..

Universidade de Aveiro Departamento de Electrónica, Telecomunicações e Informática
2012

ANDRÉ MENDES PEIXOTO

B-LIVE HOME: APLICAÇÃO MÓVEL PARA SISTEMA DE DOMÓTICA HABITACIONAL

B-LIVE HOME: MOBILE APPLICATION FOR HOME AUTOMATION SYSTEM

Universidade de Aveiro Departamento de Electrónica, Telecomunicações e Informática
2012

ANDRÉ MENDES PEIXOTO

B-LIVE HOME: APLICAÇÃO MÓVEL PARA SISTEMA DE DOMÓTICA HABITACIONAL

B-LIVE HOME: MOBILE APPLICATION FOR HOME AUTOMATION SYSTEM

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Electrónica e Telecomunicações, realizada sob a orientação científica do Doutor Alexandre Manuel Moutela Nunes da Mota, Professor Associado do Departamento de Engenharia Electrónica, Telecomunicações e Informática da Universidade de Aveiro e co-orientação científica do Mestre Paulo Jorge de Campos Bartolomeu, Assistente Convidado da Escola Superior de Tecnologia e Gestão de Águeda, Universidade de Aveiro.

COMPETE
PROGRAMA OPERACIONAL FACTORES DE COMPETITIVIDADE

QUADRO DE REFERÊNCIA ESTRATÉGICO NACIONAL
PORTUGAL 2007.2013

Fundo Europeu de Desenvolvimento Regional

FCT
Fundação para a Ciência e a Tecnologia
MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E ENSINO SUPERIOR

Aos meus pais, à minha família, aos meus amigos e professores por todo o apoio que me prestaram.

o júri

presidente

Prof. Doutor José Alberto Gouveia Fonseca
professor associado do Departamento de Engenharia Electrónica, Telecomunicações e Informática da Universidade de Aveiro

Prof. Doutor Joaquim José de Castro Ferreira
professor adjunto da Escola Superior de Tecnologia e Gestão de Águeda, Universidade de Aveiro

Prof. Doutor Alexandre Manuel Moutela Nunes Da Mota
professor associado do Departamento de Engenharia Electrónica, Telecomunicações e Informática da Universidade de Aveiro

Mestre Paulo Jorge de Campos Bartolomeu
assistente convidado da Escola Superior de Tecnologia e Gestão de Águeda, Universidade de Aveiro

agradecimentos

Quero começar por agradecer à minha família, especialmente aos meus pais que me transmitiram apoio e confiança de forma contínua, não só durante o desenvolvimento deste projecto mas durante toda a minha formação académica. Espero continuar a trabalhar cada vez mais e melhor para que um dia possa retribuir todos os esforços por eles realizados no sentido de me oferecerem as melhores condições possíveis para finalizar os meus estudos.

Aos meus orientadores no projecto, Prof. Doutor Alexandre Mota e Mestre Paulo Bartolomeu que através das suas sugestões, críticas construtivas e identificação de erros cometidos, possibilitaram melhorar o projecto em si e também alterar, para melhor, a forma como abordava determinadas situações e problemas. Tenho a certeza que todos os conhecimentos adquiridos serão uma mais-valia para o meu futuro profissional.

Agradecer também à empresa Micro I/O pela oportunidade de desenvolver este projecto nas suas instalações, permitindo ter uma visão mais realista do funcionamento do mercado de trabalho e tornando o desenvolvimento deste projecto numa experiência ainda mais gratificante. Agradecer a todas a pessoas que constituem a empresa, pois fui muito bem recebido por todos. Quero também agradecer de forma especial aos elementos que constituem o Departamento de I&D da empresa por todo o apoio prestado e pelos momentos de boa disposição durante o desenvolvimento deste projecto.

A todos os meus amigos, professores e conhecidos que tiveram influência directa ou indirecta na minha vida pessoal e académica, tanto nos bons como nos maus momentos, através dos convívios, de jantares, de aulas ou de simples conversas.

A todos os mencionados, por estes ou por outros motivos, o meu mais sincero Obrigado.

Agradeço o apoio do “Compete – Programa Operacional Factores de Competitividade”, do “QREN – Quadro Referência Estratégico Nacional” e do “FEDER – Fundo Europeu de Desenvolvimento Regional” da União Europeia no âmbito do projecto *Living Usability Lab* - Laboratório Vivo de Utilização de Tecnologias Inovadoras para as Redes de Nova Geração. Adicionalmente, agradeço também o apoio da Fundação para a Ciência e a Tecnologia no âmbito do projecto DHT-Mesh: Serviços baseados em DHT para melhorar a escalabilidade de Redes em Malha com Elevada disponibilidade (PTDC/EEA-TEL/104185/2008).

palavras-chave

Aplicação móvel, Android, domótica, interface do utilizador, experiência do utilizador, usabilidade, serviços W*eb*, socket TCP/IP.

resumo

A evolução recente no sector da computação móvel, tanto ao nível do *hardware* como do *software*, tem potenciado o aparecimento de novas tecnologias e de serviços em domínios cada vez mais amplos. Esta proliferação é suportada tanto por motivações profissionais como pessoais, permitindo o acesso a qualquer tipo de informação, em qualquer lugar e a qualquer momento.

A heterogeneidade e a flexibilidade das plataformas de desenvolvimento para dispositivos móveis existentes pode resultar em grandes variações em termos de experiência do utilizador. De modo a potenciar um "*look and feel*" homogéneo entre aplicações, existem normas e guias de boas práticas que ajudam a estabelecer referências para o seu desenvolvimento. Estes documentos abrangem conceitos tais como a interface do utilizador, experiência do utilizador, usabilidade, segurança e eficiência.

O *Living Usability Lab* é um projecto que tem como objectivo a utilização de interfaces multimodais, redes de nova geração, redes de sensores e actuadores sem-fios e computação distribuída para melhorar a qualidade de vida da população nas suas habitações, em especial, a mais idosa. Alguns dos serviços a desenvolver no âmbito deste projecto estão relacionados com a monitorização de parâmetros electro-fisiológicos e do ambiente, detecção de eventos e localização de pessoas no interior da habitação. Neste âmbito, a empresa Micro I/O, um dos parceiros do projecto, é responsável por desenvolver um sistema domótico baseado numa rede de sensores e actuadores sem-fios denominada B-Live Wireless.

Esta dissertação enquadra-se no desenvolvimento de uma aplicação móvel para controlo e monitorização do estado dos diversos sensores e actuadores do sistema B-Live Wireless. O desenvolvimento foi realizado respeitando os padrões associados a aplicações móveis, tanto ao nível do aspecto visual como da usabilidade.

keywords

Mobile application, Android, domotics, user interface, user experience, usability, web services, TCP/IP socket.

abstract

The recent developments in the field of mobile computing, both in hardware and software, boosted the emergence of new technologies and services in broad range of domains. This proliferation has been supported both by professional and personal motivations, which stemmed from the access to information, anywhere, anytime.

The heterogeneity and flexibility of the available development platforms for mobile devices can result in wide variations in terms of user experience. Standards and best practice guides help establish references for their development, allowing the promotion of a homogeneous "look and feel" among applications. These documents cover concepts such as user interface, user experience, usability, security and efficiency.

The Living Usability Lab project aims at using multimodal interfaces, next generation networks, wireless sensor and actuator networks, and distributed computing to improve the quality of life of the population in their homes, in particular, the elderly. Some of the services that will be developed in the scope of this project are related to health and environment monitoring, detecting the occurrence of events and the location of users within a house. In this context, Micro I / O, one of the project company partners, is responsible for developing a home automation system based on a wireless sensor and actuator network, named B-Live Wireless.

This thesis results from the development of a mobile application for controlling and monitoring the status of several B-Live Wireless sensors and actuators. The development was performed in order to comply with the standards of mobile applications, both in terms of visual appearance and usability.

Índice

Lista de Figuras

Lista de Tabelas

Abreviaturas e acrónimos

AC	-	Aplicação Cliente
API	-	Application Programming Interface
AS	-	Aplicação Servidora
BSD	-	Berkeley Software Distribution
CDDL	-	Common Development and Distribution License
CSS	-	Cascading Style Sheets
ESSUA	-	Escola Superior de Saúde de Aveiro
GPL	-	General Public License
HTML	-	HyperText Markup Language
HTTP	-	HyperText Transfer Protocol
IEEE	-	Institute of Electrical and Electronics Engineers
IDE	-	Integrated Development Environment
IP	-	Internet Protocol
JDBC	-	Java Database Connectivity
JDK	-	Java Development Kit
LAN	-	Local Area Network
LUL	-	Living Usability Lab
NDK	-	Native Development Kit
MPL	-	Mozilla Public License
OSI	-	Open Source Initiative
SDK	-	Software Development Kit
SOAP	-	Simple Object Access Protocol
TCP	-	Transmission Control Protocol
UI	-	User Interface
UML	-	Unified Modeling Language
UX	-	User Experience
Wi-Fi	-	Wireless Fidelity
WLAN	-	Wireless Local Area Network
WP	-	Windows Phone
WPAN	-	Wireless Personal Area Network
WS	-	Web Service
WSAN	-	Wireless Sensor/Actuator Network
WSDL	-	Web Service Description Language

XML	-	eXtensible Markup Lan

1. Introdução

1.1 Enquadramento

O crescimento verificado nos últimos anos na área da computação móvel tem vindo a potenciar o acesso a praticamente todo o tipo de informação, em qualquer lugar e a qualquer hora[1], com as pessoas a estarem cada vez mais dependentes disso, quer seja por razões profissionais ou quer por razões pessoais [1]. Profissionalmente, o acesso a informação móvel permite que, por exemplo, se possa aceder, editar e criar documentos mesmo sem estar no local de trabalho ou em casa. A nível pessoal, um cenário frequente de recurso à informação através de dispositivos móveis verifica-se no acesso às redes sociais. A computação móvel é uma área na qual também se enquadra o desenvolvimento de serviços que permitem o melhoramento das condições de vida, por exemplo, serviços de monitorização da saúde, localização ou de educação remota, entre outros [2].

No sentido de proporcionar melhores condições de vida à população, existem actualmente alguns projectos que visam desenvolver tecnologias e serviços com recurso às redes de informação móveis. Alguns destes projectos são o *Living Usability Lab*, o *Aware Home*, e o *mHealth*.

O *Aware Home* [3] é um projecto em desenvolvimento pela *Aware Home Research Iniciative* no *Georgia Institute of Technology*, desde 1998. Trata-se de um projecto de exploração multidisciplinar sobre tecnologias e serviços para habitações. Este projecto conta com a colaboração de especialistas nas áreas da saúde, educação, entretenimento, segurança e tecnologia de forma a focar a investigação em alguns problemas com impacto social e económico. Alguns serviços suportados neste projecto são: monitorização e localização de pessoas na habitação, ferramentas interactivas para promover hábitos saudáveis, interacção simplificada com elementos multimédia (por exemplo, televisão, rádio, sistema de som), partilha de conteúdos entre pessoas na habitação, entre outros.

O *mHealth* [4] é um projecto baseado no conceito da utilização de tecnologias móveis para melhorar a saúde e bem-estar em todo o mundo. Este projecto resulta um consórcio onde se incluem a *United Nations Foundation*, a *Rockfeller Foundation*, a *Vodafone Foundation*, a *HP*, a *GSM Foundation* e a *Norad*. Apesar deste conceito ser aplicado em todo o mundo, a *mHealth Alliance* tenta focar-se especialmente nos países menos desenvolvidos. Algumas das tecnologias utilizadas no âmbito do *mHealth* são o SMS e a voz em tempo real, telemedicina e navegação GPS.

O projecto *Living Usability Lab for Next Generations Networks* (LUL) [5] está a ser desenvolvido no âmbito de um projecto de Inovação e Desenvolvimento, por um conjunto de empresas (Microsoft, Micro I/O e Plux) em coordenação com algumas entidades do Sistema Científico e Tecnológico (Instituto de Engenharia Electrónica e Telemática de Aveiro, Escola Superior da Universidade de Aveiro, Faculdade de Engenharia da Universidade do Porto e

[1] Computação Ubíqua.

INESC Porto). O projecto está direccionado para o desenvolvimento de tecnologias e serviços que permitam facilitar a vida de pessoas idosas no seu quotidiano, diminuir o isolamento e a info-exclusão, aumentando a autonomia e melhorando a sua saúde e o seu bem-estar. No contexto deste projecto, está prevista a construção de um laboratório piloto para o teste das tecnologias e serviços em situações reais.

Todos estes projectos tiram partido das potencialidades oferecidas pela computação móvel de modo a monitorizar e interagir com os utilizadores, possibilitando assim um acréscimo de qualidade de vida ao nível da saúde, segurança e conforto.

No projecto LUL, a Micro I/O é responsável pelo desenvolvimento da rede de comunicações local sem fios que engloba vários sensores e actuadores. A empresa detém um conhecimento significativo nesta área devido ao desenvolvimento de uma tecnologia para domótica habitacional denominada de B-Live. Trata-se de uma rede de comunicação com fios baseada na tecnologia *Controller Area Network*, vocacionada para pessoas com limitações funcionais. Esta tecnologia foi distinguida, em 2007, com o prémio Eng. Jaime Filipe, que contempla a concepção mais inovadora e promotora da autonomia no âmbito da engenharia de reabilitação. Baseada nesta tecnologia, está a ser desenvolvida uma tecnologia semelhante, mas sem fios designada B-Live Wireless.

A Micro I/O é também responsável pelo desenvolvimento de uma interface gráfica que permita ao utilizador ter acesso à informação dos elementos que compõem o sistema, assim como permitir a mudança de estado dos actuadores. A interface gráfica foi inicialmente concebida para ser executada em *desktops* ou computadores portáteis, tendo sido decidida posteriormente a migração para plataformas móveis, o que implicou o desenvolvimento de uma aplicação móvel.

No desenvolvimento de uma aplicação móvel, existem dois factores fundamentais: o aspecto visual e as funcionalidades suportadas. Assim, se a aplicação for correctamente desenvolvida pode resultar numa utilização intuitiva e agradável além de oferecer um rendimento acrescido, factores decisivos no sucesso de qualquer produto. Embora o desenvolvimento seja um processo moroso, os futuros utilizadores esperam apenas que esta funcione segundo os requisitos técnicos que visa corresponder. Existem, por isso, regras, critérios e normas que podem e devem ser seguidas para que os aspectos citados sejam implementados correctamente na aplicação, tornando o produto mais adequado ao fim para o qual foi concebido.

Esta dissertação está enquadrada no desenvolvimento de uma aplicação móvel para um sistema de domótica habitacional. Neste sistema são disponibilizados vários serviços desenvolvidos no âmbito do projecto LUL.

1.2 Objectivos

Esta dissertação pretende descrever o processo de desenvolvimento de uma aplicação móvel para um sistema de domótica habitacional designado B-Live Wireless. Através desta aplicação móvel deve ser possível visualizar informação dos diversos dispositivos (sensores e actuadores) que compõem o sistema domótico, devendo ser possível alterar o estado de vários elementos da habitação (portas, lâmpadas, etc.) associados a dispositivos

actuadores. Adicionalmente, devem ser considerados os requisitos de usabilidade assim como o aspecto gráfico da aplicação.

1.3 Contribuições

No processo de desenvolvimento da aplicação B-Live Home, destacam-se as seguintes contribuições:

1. Análise comparativa de plataformas de *hardware* com vista à implementação de uma solução de *hardware* em que a aplicação móvel pudesse ser executada. Uma breve descrição das plataformas de *hardware* analisadas pode ser consultada no Anexo 1.
2. Análise das normas e guias de referência aplicáveis ao desenvolvimento de aplicações de *software*, principalmente vocacionados para aplicações móveis.
3. Desenvolvimento de uma aplicação Android para o pré-piloto do projecto LUL instalado no departamento de I&D da Micro I/O:
    - Desenho de ícones e imagens a serem apresentadas na aplicação;
    - Definição da disposição dos diversos elementos nos vários *layouts* que compõem a interface gráfica;
    - Definição da interacção entre os vários *layouts*;
    - Estudo das combinações de cores a apresentar na interface gráfica;
    - Adaptação da aplicação a vários tipos de ecrãs, com diferentes características (tamanho, resolução e densidade);
    - Comunicação por *socket* TCP entre a aplicação e o servidor do sistema domótico utilizando uma *Application Programming Interface* (API) proprietária da Micro I/O.
4. Desenvolvimento de uma aplicação Android para o piloto do projecto LUL instalado num laboratório presente na Escola Superior de Saúde da Universidade de Aveiro:
    - Desenho de ícones e imagens a serem apresentadas na aplicação;
    - Definição da disposição dos diversos elementos nos vários *layouts* que compõem a interface gráfica;
    - Definição da interacção entre os vários *layouts*;
    - Estudo das combinações de cores a apresentar na interface gráfica;
    - Adaptação da aplicação a vários tipos de ecrãs, com diferentes características (tamanho, resolução e densidade);

- Comunicação entre a aplicação e o servidor do sistema domótico com recurso a *Web Services*.

1.4 Estrutura da dissertação

Este documento está divido em cinco capítulos principais. No primeiro é apresentado o enquadramento da dissertação, os objectivos a atingir e as contribuições resultantes do trabalho.

No segundo capítulo é apresentado o estado da arte associado ao desenvolvimento de aplicações móveis, sendo abordadas as plataformas de desenvolvimento de aplicações móveis, normas e boas práticas, sendo ainda apresentado o conceito de projecto *open source*.

O terceiro capítulo introduz o trabalho anterior realizado no âmbito do projecto LUL. Neste sentido, é feita uma descrição geral do projecto, abordando o sistema B-Live Wireless ao nível da arquitectura de *hardware* e de *software*.

O processo de desenvolvimento da aplicação B-Live Home é apresentado no capítulo quatro, sendo descrito o enquadramento da aplicação no sistema B-Live Wireless e são expostos os requisitos funcionais e não funcionais que a aplicação possui. Adicionalmente, é apresentada a definição da aplicação (diagrama de *use cases* e *mockup*), passando-se de seguida para a implementação ao nível das imagens, ícones e *layouts*. Posteriormente, é descrita a interacção entre os *layouts* e a comunicação com o servidor do sistema.

No quinto e último capítulo, são apresentadas as conclusões do trabalho e discutidos os aspectos a melhorar no futuro.

2. Estado da arte

Neste capítulo é apresentado o estado da arte associado ao desenvolvimento de aplicações móveis, sendo abordadas algumas plataformas de desenvolvimento de aplicações móveis, normas e boas práticas. Finalmente é apresentado o conceito de projecto *open source*.

2.1 Conceitos básicos

Para se perceber melhor a utilidade das interfaces gráficas, é importante considerar alguns conceitos básicos que são utilizados nesta área, nomeadamente:

- **Interface do utilizador (UI – *User Interface*):** Este termo é utilizado quando pretende realçar o aspecto visual e as funcionalidades da aplicação. Por outras palavras, é o modo como um indivíduo interage com um dispositivo electrónico, através de menus, ícones, janelas, linha de comandos ou atalhos de teclado. Correspondem aos métodos utilizados para que o utilizador possa usufruir da aplicação, interagindo assim com o sistema [6].
- **Experiência do utilizador (UX – *User Experience*):** Este conceito é algo mais abrangente pois engloba o UI e o "comportamento" da aplicação, bem como as sensações que os utilizadores conseguem extrair do uso da aplicação. Não é suficiente a aplicação ser atraente visualmente, também deve proporcionar satisfação aos utilizadores [6].
- **Usabilidade (*Usability*):** A usabilidade de uma aplicação está relacionada com a facilidade de aprendizagem e utilização do produto. Também é tida em conta a eficácia, eficiência, a memorização[2], erros possíveis de serem provocados pelos utilizadores e satisfação no uso do produto [7]. Existem empresas que fornecem serviços de teste de usabilidade, por exemplo a Usernomics [8].
- **Intuitivo:** Quando se refere que uma interface gráfica deve ser intuitiva significa que esta deve ser de fácil aprendizagem, não sendo necessário um período longo de adaptação/treino para poder utilizar a maior parte das suas funcionalidades. No entanto, existem especialistas [9] que preferem usar o termo "familiar" como sinónimo de intuitivo, isto é, uma interface gráfica é mais intuitiva se usar mecanismos que o utilizador já conhece de outros sistemas. Por exemplo, o símbolo "||" é reconhecido por todos como o símbolo que significa "pausa" em sistemas de leitura multimédia. De notar

[2] Após um longo período sem usar a aplicação, o grau de facilidade de voltar a utilizar sem problemas.

que se trata apenas de uma convenção e que só pelo símbolo não se conseguiria perceber o significado intuitivamente.

2.2 Plataformas de desenvolvimento

As aplicações gráficas devem ser projectadas tendo em conta os dispositivos em que vão ser utilizadas, bem como os seus utilizadores. Por exemplo, não é recomendável oferecer uma experiência iPhone a um utilizador de BlackBerry, pois são duas realidades diferentes, resultando num período de adaptação indesejável [10]. Esta é uma das justificações para que cada plataforma tenha a sua estratégia de usabilidade e de desenvolvimento de aplicações.

De seguida, são apresentados aspectos relevantes relativos a algumas das plataformas de desenvolvimento de aplicações móveis actualmente mais populares: Windows Phone [11], iOS [12] e Android [13]. Para além destas, são também abordadas ferramentas de desenvolvimento multiplataforma.

2.2.1 Windows Phone

O Windows Phone (WP) é o mais recente sistema operativo móvel desenvolvido pela Microsoft [14] e trata-se do sucessor do Windows Mobile, não sendo, no entanto, compatível com este. Lançado na segunda metade de 2010, apresenta um *user interface* completamente remodelado comparativamente com o seu antecessor, tanto a nível gráfico, como ao nível dos ecrãs panorâmicos, possuindo também melhoramentos significativos no que toca à experiência multimédia (música, vídeo e jogos) [15]. Em Fevereiro de 2011, a Microsoft e a Nokia [16] anunciaram uma parceria no sentido de tornar o WP no sistema operativo principal dos dispositivos da Nokia. Esta parceria tinha como objectivo competir com os restantes sistemas operativos móveis presentes no mercado, sendo já uma das plataformas mais atractivas, tanto ao nível do programador como do utilizador [17]. A última versão disponível é o Windows Phone 8.

As aplicações para WP podem ser programadas em várias linguagens, entre as quais se destacam o C# e o Visual Basic [18]. A Microsoft fornece diversas ferramentas de desenvolvimento de aplicações como é o caso das *frameworks*[3] XNA e Silverlight. A XNA é utilizada essencialmente na produção de jogos de elevado desempenho enquanto a Silverlight é mais usado em aplicações gráficas 2D. A Microsoft disponibiliza também um SDK (*Software Development Kit*) que contém os seguintes componentes [11]:

- Visual Studio 2010 Express para WP;
- Emulador de Windows Phone;
- XNA Game Studio;

[3] Conjunto de bibliotecas e classes de *software* reutilizáveis.

- Expression Blend para WP;
- Exemplos;
- Documentação;

Os sistemas operativos suportados pelo SDK são o Windows Vista com o *Service Pack* 2 e o Windows 7, tanto nas versões 32-bits como 64-bits. O Windows XP e Windows Server não são suportados, assim como máquinas virtuais.

A licença do WP pertence à Microsoft pelo que o código fonte se encontra como código fechado. Isto implica o uso deste *software* de acordo com certas condições que impõem restrições no que diz respeito à modificação do *software* e futura redistribuição, por exemplo.

Relativamente ao processo de desenvolvimento de aplicações, existem alguns passos que são necessários percorrer de forma sequencial, e que vão desde a concepção da ideia da aplicação até à sua disponibilização no Marketplace (serviço de disponibilização de aplicações para Windows Phone desenvolvidas por terceiros) [19]. De seguida é apresentada uma ilustração simplificada do processo de desenvolvimento de aplicações em WP.

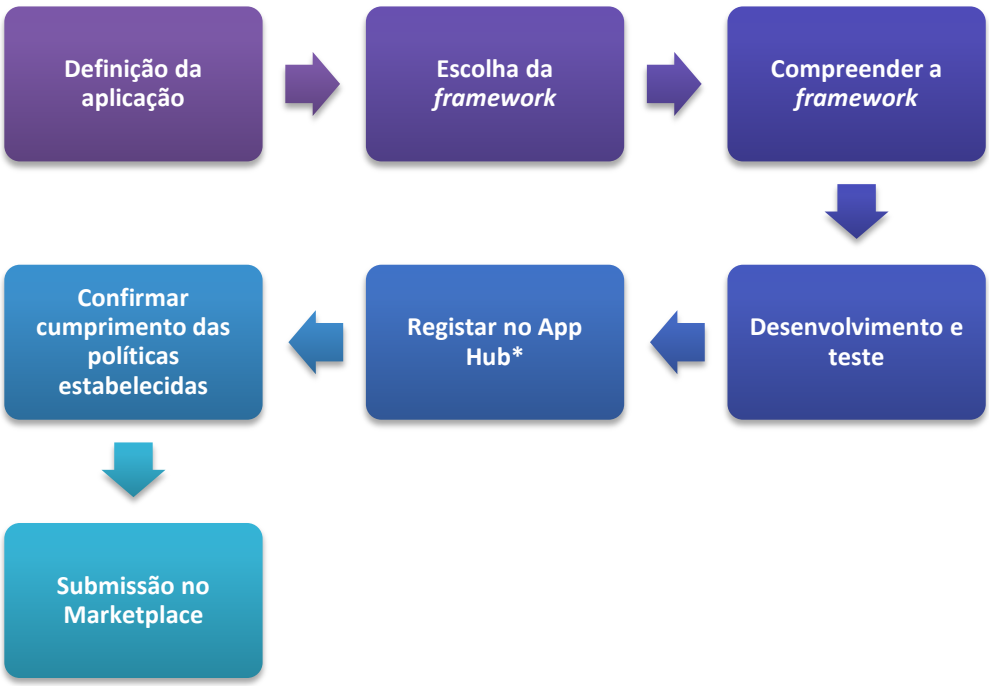


Figura 1 - Processo para desenvolvimento de aplicações em Windows Phone

* Serviço de suporte ao desenvolvimento e distribuição de aplicações

O desenvolvimento nesta plataforma é gratuito. Contudo, se for pretendido distribuir as aplicações no Marketplace é necessário pagar uma subscrição anual de $99 USD, sendo possível submeter um número ilimitado de aplicações pagas, enquanto aplicações gratuitas existe um limite de 100 aplicações. Submissões adicionais têm o custo de $19,99 USD por aplicação. O preço mínimo para aplicações pagas é de $0,99 USD e o máximo é de $499,99 USD, com a Microsoft a receber 30% desse valor.

Os guias de referência para o *design* dos UI são apresentados no sentido de ajudar a estruturar o modelo de interacção com o utilizador, para que seja possível oferecer aos utilizadores finais da aplicação a melhor UX. Estes guias abordam aspectos como a navegação, o botão de retroceder (*back button*), a *application bar*, a orientação do ecrã, os ícones e imagens, os temas, configurações da aplicação, modo de toque, as fontes do texto e o teclado no ecrã.

2.2.2 Apple iOS

O iOS é o sistema operativo móvel desenvolvido pela Apple [20] e foi lançado em 2007. Ao contrário de outras plataformas como o WP e o Android que, através de parcerias com outras empresas possibilitam o uso da plataforma em diversos dispositivos, este sistema operativo apenas pode ser instalado e utilizado no *hardware* disponibilizado pela Apple. Os dispositivos que suportam o iOS são o iPod Touch, o iPhone, o iPad e a Apple TV. Todos estes dispositivos têm já várias versões de *hardware* (por exemplo, iPhone 3G, iPhone 4, iPad 2, iPad 3). A última versão estável deste sistema operativo corresponde ao iOS 5.1.1.

Ao contrário do que sucedia as versões inicias deste sistema operativo, em que apenas era permitido programar para iOS em Objective-C, C++ e C, actualmente é possível programar nesta plataforma em outras linguagens recorrendo a ferramentas específicas capazes de interpretar e traduzir o código de modo a que o sistema operativo possa executar a aplicação. Exemplos deste tipo de ferramentas serão apresentados na secção 2.2.4 deste documento. Ainda assim, a linguagem de programação principal continua a ser o Objective-C e é nesta linguagem que se consegue obter o melhor desempenho, principalmente devido ao facto da Apple fornecer *frameworks* também em Objective-C [21].

A Apple disponibiliza um SDK completo que permite desenvolver aplicações para executar em Mac, iPhone e iPad. Um dos requisitos para se desenvolver aplicações para iOS de forma nativa[4] é ser necessário possuir um Mac a executar o sistema operativo OS X Snow Leopard ou superior. O outro requisito é estar registado como *Apple Developer* no *site* da Apple. O iOS SDK vem integrado com as *frameworks* Cocoa e CocoaTouch e é composto essencialmente por:

- Xcode IDE;
- Compilador Apple LLVM;
- Instruments;
- Simulador iOS;

Tal como acontece no caso do WP, o iOS também é *software* proprietário, neste caso, da Apple e o código fonte também é fechado. O processo de desenvolvimento para esta plataforma, até à fase de distribuição na AppStore (serviço de disponibilização de aplicações para iOS desenvolvidas por terceiros) [22], assemelha-se, de certo modo, ao processo apresentado para o WP. Contudo, a Apple propõe também a estrutura organizacional de uma equipa para o desenvolvimento de uma aplicação. Na Figura 2 está apresentada uma perspectiva de alto nível deste processo.

[4] Sem recorrer a máquinas virtuais.

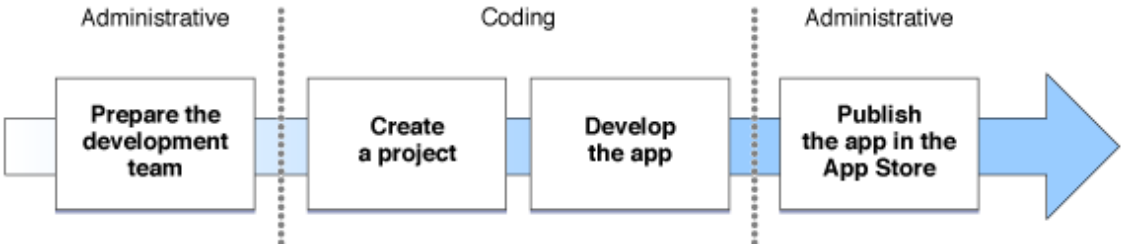


Figura 2 - Processo para desenvolvimento de aplicações em iOS

Fonte: *iOS Dev Center* [12]

Embora estes passos estejam apresentados numa ordem lógica, pode ser necessário percorrer estas fases por outra ordem. Estas fases estão classificadas como sendo "Administrativas" ou de "Código" em que no primeiro caso as funções são desempenhadas por alguém da equipa com responsabilidades legais[5], e no segundo pelos programadores inseridos na equipa. Na última fase, o processo de publicação da aplicação implica a aprovação prévia nos testes realizados pela Apple e só depois é que a aplicação é disponibilizada na AppStore. O preço da aplicação é atribuído pelo desenvolvedor da aplicação, do qual a Apple recebe 30% desse valor por cada *download* efectuado.

Relativamente aos custos de desenvolvimento, variam de acordo com o programa escolhido. A Apple disponibiliza 3 programas de desenvolvimento. Na Tabela 1 estão apresentadas as características dos programas.

Tabela 1 – Características dos programas de desenvolvimento para iOS

Programa	Preço (subscrição anual)	Descrição
iOS Developer Program	$99 USD	Modalidade geral para desenvolvimento individual ou em ambiente empresarial
iOS Developer Enterprise Program	$299 USD	Desenvolvimento de *software* proprietário para uso interno de uma empresa ou organização
iOS Developer University Program	Gratuito	Desenvolvimento no âmbito académico

O teste e a execução das aplicações em dispositivos físicos (por exemplo, iPhone, iPad) implica a adesão a um dos programas apresentados na tabela anterior, caso contrário, a execução destas está limitada ao simulador integrado com o SDK. A disponibilização de aplicações na AppStore é possível apenas através da adesão do programa "*iOS Developer Program*".

A Apple disponibiliza também uma extensa lista de guias de referência e princípios para o *design* da interface no sentido de ajudar no desenvolvimento da UI e UX da aplicação. Nesta lista constam aspectos como os gestos que o utilizador pode realizar quando toca no

[5] Denominado de *Team Agent.*

ecrã, consistência entre as aplicações, métodos de *feedback* para o utilizador, diferenciação entre *design* para iPhone e para iPad, entre muitos outros aspectos.

2.2.3 Android

O Android é uma plataforma de desenvolvimento para dispositivos móveis, com origem num consórcio de organizações denominado de Open Handset Alliance [23], cuja liderança está a cargo da Google. O objectivo deste consórcio através do projecto "Android Open Source Project" é desenvolver um produto que melhore a experiência dos utilizadores finais na utilização de dispositivos móveis. Uma das particularidades mais interessantes desta plataforma é o facto de ser *software open source* [24]. A definição de *open source* e as suas implicações são apresentadas no capítulo 2.4 deste documento. O Android é baseado em Linux e apesar de ser bastante utilizado em dispositivos móveis como *smartphones* e *tablets*, também pode ser usado outro tipo de dispositivos como é o caso de sistemas embutidos mais tradicionais[6]. Isto deve-se ao facto desta plataforma ser altamente configurável. No Anexo 1 são apresentadas algumas plataformas de *hardware* para desenvolvimento em Android. A última versão estável do Android disponível nesta data é o Jelly Bean (4.1).

A principal linguagem de programação utilizada no desenvolvimento de aplicações para Android é o Java, com cada aplicação a ser executada no seu próprio processo, com uma instância para uma máquina virtual[7] denominada de "*Dalvik Virtual Machine*". Para além da linguagem Java, também é possível desenvolver para Android em linguagem C e C++, recorrendo para isso ao *Android Native Development Kit* (NDK), disponibilizado pela Google. Este NDK possibilita ao programador produzir partes do código da aplicação que sejam sensíveis ao desempenho, necessitando de elevada performance [25].

A Google também disponibiliza um SDK que contém os seguintes componentes:

- SDK Tools;
- SDK Platform-tools;
- Plataformas Android;
- Driver USB para Windows;
- Exemplos;
- Documentação;

Este SDK pode ser integrado no Eclipse IDE [26] recorrendo ao ADT-plugin, sendo esta a forma mais fácil e rápida de começar a desenvolver aplicações para Android. Através do Eclipse é possível executar as aplicações num emulador, facilitando desta forma o processo de desenvolvimento. Um dos requisitos prévios para a instalação do SDK é ter instalado a versão 6 ou superior do *Java Development Kit* (JDK).

Como já foi referido, o projecto Android é *open source* e a maioria do código fonte é disponibilizado sob a licença Apache 2.0. No capítulo 2.4 deste documento é apresentado as implicações derivadas do facto do *software* ser libertado sob a licença Apache 2.0.

[6] Sistemas que realizam tarefas específicas, optimizados para possuírem consumos reduzidos, tamanhos reduzidos e desempenhos elevados.
[7] Máquina implementada em *software* capaz de executar outros programas.

O desenvolvimento de aplicações em Android é bastante facilitado recorrendo às ferramentas disponíveis no SDK, sendo o processo de desenvolvimento proposto pela Google apresentado na Figura 3.

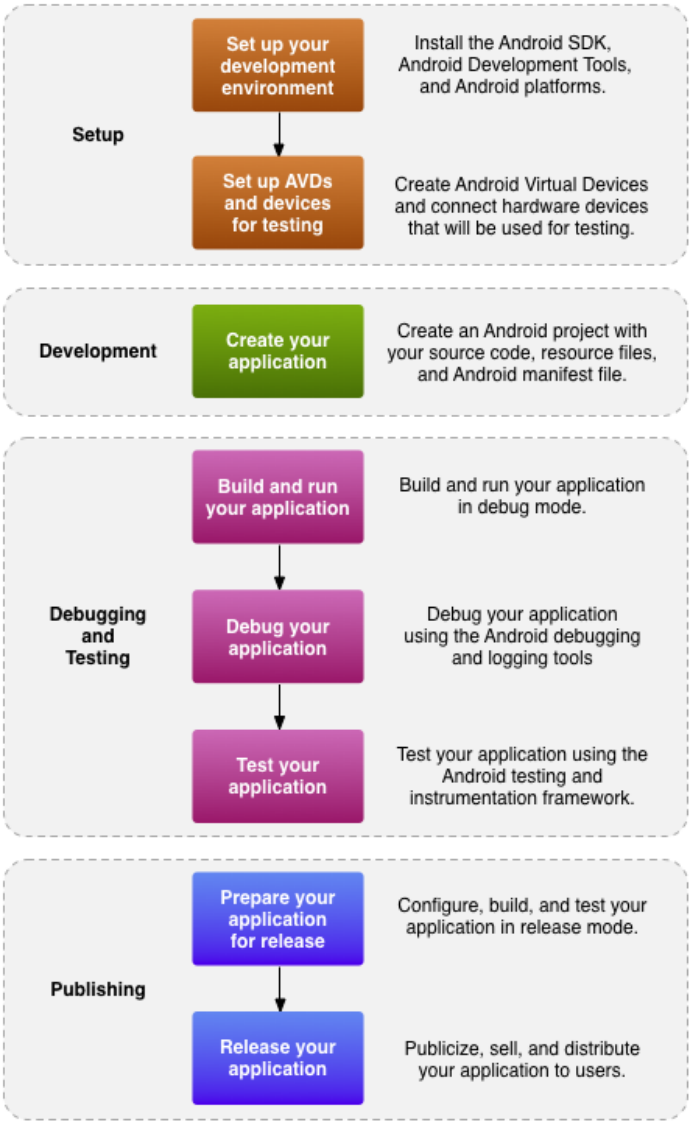


Figura 3 – Processo de desenvolvimento de aplicações em Android

Fonte: *Android Developers* [13]

O serviço de disponibilização de aplicações da Google é denominado de Google Play e é através deste serviço que os programadores publicam as suas aplicações. Para ser possível publicar as aplicações neste serviço, é necessário o efectuar o registo mediante o pagamento único de uma taxa de $25 USD. É, no entanto, é possível desenvolver, executar e testar as aplicações tanto no emulador como em dispositivos físicos, sem qualquer custo adicional. Tal como nas outras plataformas analisadas, a percentagem recebida pela Google é de 30% do valor da aplicação, sendo os restantes 70% recebidos pela entidade que a desenvolveu. Contudo, ao contrário do que sucede para Windows Phone e iOS, as aplicações para Android não estão sujeitas a um processo de aprovação para submissão no Google Play.

A Google propõem também alguns guias de referência para o design da UI e da UX, no sentido de o programador integrar da melhor forma a sua aplicação no restante sistema operativo. Estes guias vão sendo actualizados de acordo com a última versão lançada do sistema operativo. Os guias abordam aspectos como os ícones a apresentar, o tipo de letra, cores, métricas, temas, *widgets*[8], gestos, navegação, notificações, entre outros.

2.2.4 Ferramentas Multiplataforma

Por vezes é necessário desenvolver a mesma aplicação para diferentes plataformas num curto espaço de tempo. Contudo, o desenvolvimento da aplicação para cada sistema operativo individualmente pode ser mais dispendioso e mais demorado. Estes aspectos podem ser minimizados recorrendo a ferramentas de desenvolvimento multiplataforma, pois estas permitem desenvolver uma aplicação capaz de executar em vários sistemas operativos. Por exemplo, recorrendo a uma destas ferramentas, é possível desenvolver de uma vez uma aplicação que pode executar num dispositivo com sistema operativo Android ou num com iOS.

Neste âmbito, existem várias ferramentas deste tipo tais como o PhoneGap, Corona, DragonRAD, RhoMobile, Appcelerator, MoSync, entre outros. De seguida é apresentado um resumo das características de três destas soluções.

PhoneGap

O PhoneGap [27] é uma ferramenta multiplataforma *open source* desenvolvida originalmente pela Nitobi Software. Esta empresa foi adquirida, em Outubro de 2011, pela Adobe [28]. Após esta aquisição a licença utilizada pelo PhoneGap é o Apache 2.0, o que significa que os programadores podem utilizar esta *Framework* para aplicações móveis que são gratuitas, comerciais, *open source* ou qualquer combinação destas hipóteses. O PhoneGap utiliza tecnologias *Web* tais como JavaScript, HTML e CSS3, de modo a efectuar a ligação entre as aplicações *Web* e os dispositivos móveis. As plataformas suportadas são o iOS, Android, Windows Phone, Blackberry OS [29], WebOS [30], Symbian e Bada [31].

RhoMobile

O RhoMobile [32] é uma solução *open source* desenvolvida pela RhoMobile Inc. e lançada em 2008. Este *software* é disponibilizado sob a licença MIT (Massachusetts Institute of Technology). Em Outubro de 2011 a RhoMobile foi adquirida pela Motorola Solutions. A linguagem de programação utilizada por esta *framework* é o Ruby, uma linguagem dinâmica e *open source,* focada na simplicidade e produtividade. Contudo, é também possível programar em outras linguagens como HTML ou JavaScript. O padrão de desenvolvimento é o *Model-View-Controller* (MVC). O RhoMobile é composto por um conjunto de produtos, cada um com a sua funcionalidade: Rhodes (Desenvolvimento), RhoConnect (Integração), RhoHub (Disponibilização) e RhoGallery (Gestão). As plataformas suportadas são o Android, iOS, Windows Phone, Windows Mobile e Blackberry OS.

[8] Funcionalidade que permite visualizar pequenas quantidades de informação relativa a uma aplicação, no *home screen* do sistema.

MoSync

A MoSync AB [33] é uma empresa de *software* fundada em 2004 em Estocolmo, Suécia. Esta empresa está direccionada para a produção de SDKs para desenvolvimento de aplicações móveis, disponibilizando até ao momento dois produtos: o MoSync SDK e o MoSync Reload. O MoSync SDK permite desenvolver aplicações em C/C++ mas também com HTML5 e JavaScript, enquanto o MoSync Reload permite editar aplicações móveis em HTML5, JavaScript e CSS e observar em tempo real o efeito das alterações, quer seja no emulador ou no dispositivo físico. As plataformas suportadas são o Android, Blackberry OS, iOS, JavaME, Moblin, Symbian, Windows Mobile e Windows Phone. Estes produtos estão sob dois tipos de licenças: GPLv2[9] e comercial. Isto permite ao desenvolvedor decidir se pretende desenvolver *software* livre ou proprietário.

2.2.5 Análise comparativa

Através das análises efectuadas às plataformas de desenvolvimento de aplicações móveis nativas (Windows Phone, iOS e Android), pode-se verificar que tanto existem alguns aspectos comuns como diferenças assinaláveis entre elas. Em comum tem-se, por exemplo, o facto de todas as plataformas oferecerem um serviço para disponibilização de aplicações desenvolvidas por terceiros, embora com diferentes condicionantes, mas que permitem alargar o espectro de utilizadores tanto ao nível do desenvolvimento como ao nível dos utilizadores finais. Quanto aos aspectos diferenciadores, tem-se, por exemplo, as linguagens de programação principais a utilizar em cada plataforma, assim como o suporte para *hardware* que no caso do iOS está restringido aos produtos da Apple.

De seguida, recorrendo à Tabela 2, pretende-se apresentar algumas das características das plataformas de desenvolvimento nativas analisadas.

[9] Breve descrição no capítulo 2.4.

Tabela 2 – Tabela comparativa das plataformas de desenvolvimento de aplicações móveis

	Windows Phone	iOS	Android
Organização	Microsoft	Apple	Google/Open Handset Alliance
Open source	Não	Não	Sim
Linguagem de programação principal	C#	Objective-C	Java
IDE	Visual Studio 2010	Xcode	Eclipse
Multitarefa	Só a partir da versão 7.1	Só a partir da versão 4.0	Sim
Emulador	Sim	Sim	Sim
Dispositivos suportados	*Smartphones* e *tablets* (Nokia, Samsung, HTC, entre outros)	iPhone, iPad, iPod Touch	*Smartphones* e *tablets*, KITs de *hardware* (Asus, Samsung, Motorola, entre outros)
Site de apoio aos desenvolvedores	Sim[11]	Sim[12]	Sim[13]
Serviço de disponibilização de aplicações	Marketplace	AppStore	Google Play
Processo de aprovação das aplicações	Sim	Sim	Não
Custo de desenvolvimento	$99 USD/Ano (para poder disponibilizar no Marketplace)	$99 USD/Ano (para poder disponibilizar na AppStore)	$25 USD (para poder disponibilizar no Google Play)
Tipo de licença	Proprietária	Proprietária	Apache 2.0
Número de aplicações disponíveis [34]	Cerca de 70,000	Cerca de 600,000	Cerca de 400,000
Quota de mercado[35]	1.9%	23.8%	50.9%

A tabela anterior permite identificar alguns aspectos que diferem entre as plataformas e que devem ser tidos em consideração aquando da escolha da plataforma a utilizar. Aspectos como o *software* ser *open source* ou não, a linguagem de programação, os dispositivos físicos suportados, os custos de desenvolvimento, ou o tipo de licença, podem influenciar a decisão.

Como foi referido, outra solução para o desenvolvimento de aplicações móveis são as ferramentas multiplataforma. Apesar de em certos casos ser considerada uma boa opção[10], noutros pode apresentar vários inconvenientes[11] que é necessário ter em consideração antes de se optar por este tipo de solução. Na tabela seguinte, é apresentada uma comparação

[10]Por exemplo, quando é necessário um período de desenvolvimento reduzido (para lançar para o mercado), para as diversas plataformas.
[11] Por exemplo, pode não proporcionar ao utilizador a UX espectável.

entre as soluções de desenvolvimento nativas em oposição às soluções multiplataforma, tendo em consideração as vantagens e desvantagens de cada uma delas [36].

Tabela 3 – Vantagens e desvantagens das plataformas nativas *versus* multiplataformas

	Nativas	Multiplataforma
Vantagens	• Estável; • Existência de guias de referência para o UI; • Melhor resultado do UI, pois aproveita melhor as características do ecrã; • Apoio técnico directo; • *Frameworks* actualizadas;	• Solução *open source;* • Conjunto de linguagens de programação que serve para todas; • Desenvolvimento rápido;
Desvantagens	• Linguagens de programação diferentes; • Diferentes padrões de *design* do UI; • Tempo de desenvolvimento mais elevado;	• Instável; • Apoio técnico limitado; • APIs limitadas; • *Frameworks* podem estar desactualizadas;

Finalmente, pode-se concluir o seguinte relativamente a cada uma das plataformas nativas apresentadas:

- **Windows Phone:** Trata-se da plataforma móvel mais recente de entre as 3 analisadas, pelo que ainda não tem uma representação muito significativa no mercado. O facto de não ser *open source,* do número ainda limitado de dispositivos que integram esta plataforma e os custos associados ao desenvolvimento podem ser aspectos que abrandam o interesse por desenvolver para WP.
- **iOS:** O iOS é uma referência ao nível do desenvolvimento de aplicações móveis, sendo a consistência entre as aplicações um dos seus pontos fortes, devido ao processo rigoroso de avaliação que antecede a publicação na AppStore. Em contrapartida, para além das razões apresentadas para o WP, a falta de flexibilidade no *hardware*, as limitações de customização das aplicações, e a linguagem de programação ser de mais baixo nível (Objective-C)que o Java e o C# podem tornar o desenvolvimento para esta plataforma num processo complexo.
- **Android:** O Android possui uma quota do mercado móvel muito significativa. Um dos pontos menos positivos desta plataforma (dependendo da perspectiva) prende-se com a existência de uma grande diversidade tanto a nível de *hardware* como de *software*. Isto resulta em algumas inconsistências nesta plataforma, por exemplo, no que ao UI e UX diz respeito. Contudo, a plataforma Android apresenta vantagens muito interessantes relativamente às outras plataformas: é *open source*, tem custos de desenvolvimento relativamente reduzidos, é bastante configurável e possui a capacidade de executar em vários tipos de *hardware*, o que permite, por exemplo, criar

soluções de *hardware* inovadoras para sistemas embutidos que executem Android.

2.3 Introdução aos padrões utilizados no *design* de *software*

As interfaces gráficas devem possuir a melhor integração possível com os restantes componentes do sistema no qual estão inseridas, tanto a nível visual como ao nível da usabilidade. Por isso, têm sido estabelecidos vários padrões que normalizam estes aspectos, para que existam pontos de referência que garantam, a quem pretenda desenvolver novas aplicações o possa fazer com a certeza que estas serão integradas da melhor forma no sistema.

Além de normas genéricas, algumas empresas de referência assumem a definição de guias de boas práticas no que toca a usabilidade, UX e UI [37]. Por exemplo, a Apple possui as suas próprias *guidelines* para o iPhone, iPad e Mac OS X, assim como a Microsoft para o Windows XP, Windows Seven e Windows Phone, a Google, para dispositivos Android ou a HP para o Palm webOS. Existem também boas práticas gerais que devem ser seguidas no desenvolvimento de novas aplicações.

Seguidamente, são apresentadas algumas normas importantes, assim como algumas boas práticas usadas na elaboração de novas aplicações.

2.3.1 Normas

As normas incluem toda a informação relacionada com *layouts* gráficos, janelas, menus, ícones, formas, tamanhos, cores, comportamento interactivo da aplicação, desenvolvimento, teste e documentação. Por isso, a escolha acertada do conjunto de normas a utilizar no desenvolvimento de uma interface gráfica é um processo muito importante que depende muito da plataforma e do tipo de dispositivo alvo.

De seguida são apresentados alguns *standards*, com uma breve descrição do que é abordado em cada um deles.

Norma ISO 9126

O *standard* ISO 9126 (*Software engineering — Product quality*) [38] que data de 1991 está vocacionado para a qualidade do *software*. Define seis características relativas à qualidade: funcionalidade, confiabilidade, usabilidade, eficiência, portabilidade e manutenção. Cada uma destas características está associada a uma questão principal, como se encontra apresentado na Figura 4.

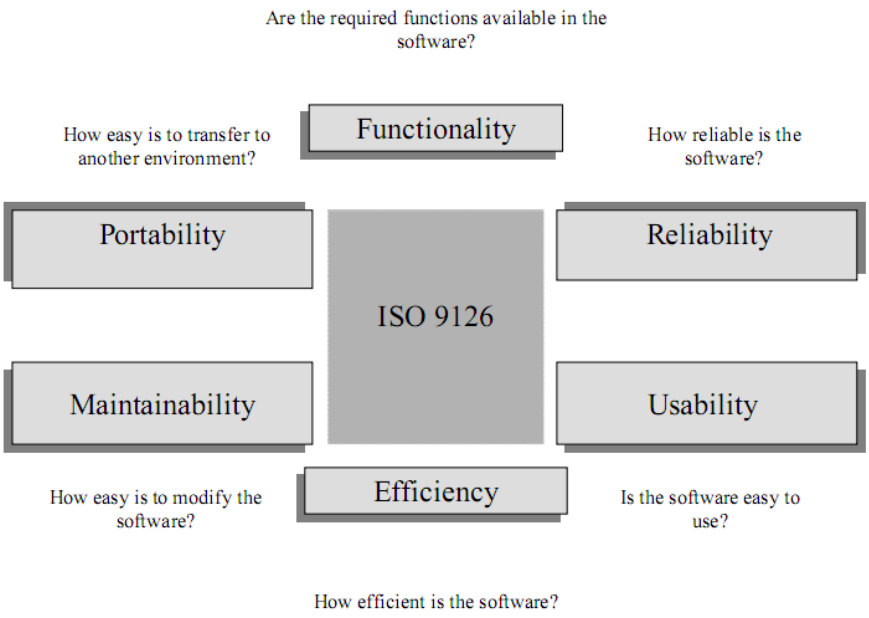


Figura 4 – Características da Norma ISO 9126

Fonte: *Applying the ISO 9126 model to the evaluation of an elearning system* [39]

Estas características encontram-se subdivididas em sub-características (Tabela 4), que representam um modelo detalhado para a avaliação de qualquer *software*. Esta norma define também as métricas usadas na medição da qualidade destes atributos.

Tabela 4 - Características e sub-características avaliadas pela norma ISO 9126

Característica	Sub-característica	Explicação
Funcionalidade	Adequabilidade	O *software* consegue executar as tarefas requeridas?
	Exactidão	O resultado é o esperado?
	Interoperabilidade	O sistema consegue interagir com outro sistema?
	Segurança	O *software* gere acessos não autorizados?
Fiabilidade	Maturidade	A maioria das falhas foi eliminada ao longo do tempo?
	Tolerância a falhas	O *software* é capaz de manipular erros?
	Recuperação	Após uma falhar, o *software* é capaz de voltar a funcionar e restaurar os dados?
Usabilidade	Compreensibilidade	O utilizador compreende como utilizar o sistema facilmente?
	Capacidade de aprendizagem	O utilizador consegue aprender facilmente a utilizar o sistema?
	Operatividade	O utilizador consegue utilizar o sistema sem muito esforço?
	Atractividade	A interface tem bom aspecto?
Eficiência	Comportamento temporal	Quão rapidamente o sistema responde?
	Utilização de recursos	O sistema utiliza os recursos eficientemente?
Manutenção	Análise	As falhas conseguem ser diagnosticadas facilmente?
	Mutabilidade	O *software* pode ser modificado facilmente?
	Estabilidade	O *software* continua a funcionar depois de modificado?
	Teste	O *software* pode ser testado facilmente?
Portabilidade	Adaptação	O software pode ser movido ara outros cenários?
	Instalação	O software é instalado facilmente?
	Conformidade	O software está conforme as normas de portabilidade?
	Substituição	O software pode ser substituído por outro facilmente?
Todas as características	Correspondência	O software está de acordo com as leis e regulamentos?

Fonte: Adaptado de "*Applying the ISO 9126 model to the evaluation of an elearning system*" [39]

Norma ISO 9241-11

Esta norma foi publicada em 1998 [40] e deriva da norma ISO 9241 (*Ergonomic requirements for Office work with visual display terminals* (VDTs)), que se encontra dividida em várias partes, correspondendo a ISO 9241-11 à parte 11. No

Anexo 2 é possível verificar a lista dos componentes abrangidos por esta norma. A norma ISO 9241-11 está estritamente relacionada com o termo usabilidade e utiliza o ponto de vista de utilizadores específicos para atingir objectivos tais como eficácia, eficiência e satisfação num contexto específico de uso. Nesta norma é possível encontrar os métodos usados para especificar e medir a usabilidade dos produtos. Na Figura 5 é indicada a informação necessária para a especificação e medida da usabilidade.

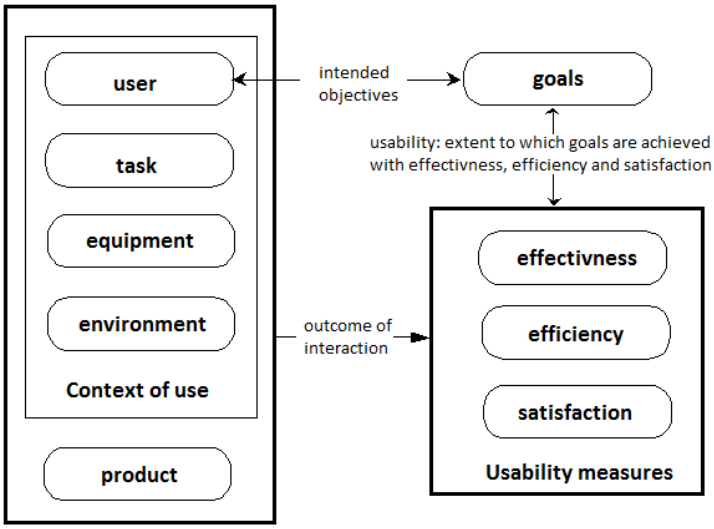


Figura 5 – Informação necessária para a especificação e medida da usabilidade

Fonte: *ISO/IEC 9241-11Ergonomic requirements for office work with visual display terminals (VDT)s* [40]

Através da Figura 5 é possível observar que para a especificação e medida da usabilidade é necessária a seguinte informação:

- A descrição dos objectivos a atingir que se visa atingir;
- A descrição dos contextos de utilização tais como os utilizadores, tarefas, equipamento e cenários. Os aspectos relevantes dos contextos e o nível de detalhe exigidos na descrição dependem do âmbito das questões a serem abordadas.
- Valores pretendidos ou actuais de eficácia, eficiência e satisfação para os contextos projectados;

Norma ISO/IEC 11581-1

AISO/IEC 11581-1 (2000) [41] é uma das componentes da ISO/IEC 11581 (*Information technology – User system interfaces and symbols – Icons symbols and functions*). No

Anexo 2 é possível ver a lista das componentes que constituem esta norma. A ISO/IEC 11581-1 fornece um conjunto de indicações necessárias para o desenvolvimento e *design* de ícones e as suas aplicações nos *displays*. Na Figura 6 são representados os passos gerais usados no desenvolvimento de ícones para que seja possível ao utilizador interpretar o ícone e associá-lo a uma determinada função.

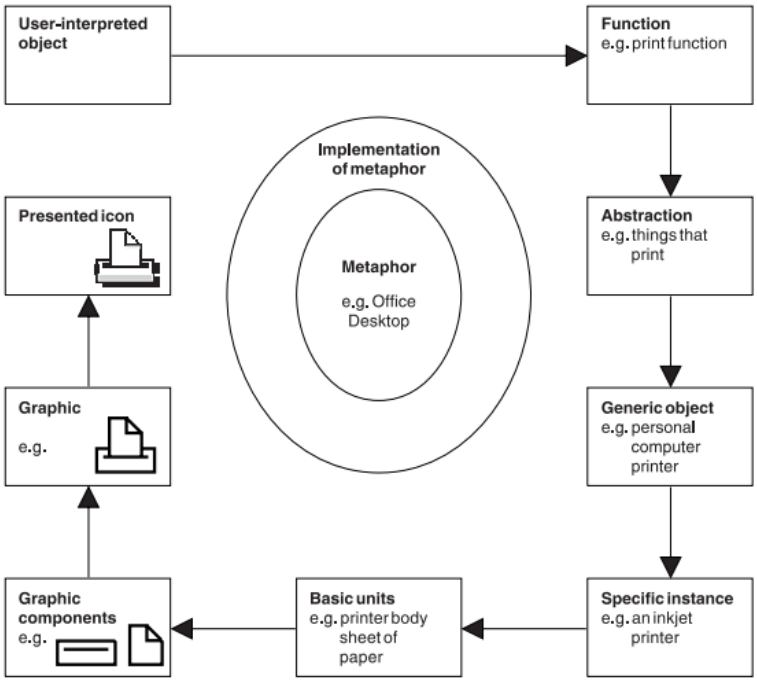


Figura 6 – Especificação de ícones

Fonte: *ISO/IEC11581-1 Information technology - User system interfaces and symbols - Icons symbols and functions* [41]

Através da imagem é possível observar os seguintes conceitos utilizados para o desenvolvimento de ícones:

- *Metaphor*: o ícone fornece uma ligação visual entre uma função e um objecto num ambiente metafórico;
- *Function*: função do sistema computacional representado pelo ícone;
- *Abstraction*: representação conceptual da funcionalidade em termos de classes de objectos;
- *Generic object*: classe particular do objecto que pode executar a funcionalidade;
- *Specific instance*: concepção do objecto genérico que corresponde a um determinado objecto particular no ambiente metafórico;
- *Basic units*: elementos do objecto particular susceptíveis de serem representados graficamente;
- *Graphic components*: representação gráfica das *basic units* necessárias para a criação do ícone;

- *Graphic*: representação gráfica do símbolo que se pretende representar como ícone;
- *Presented icon*: ícone apresentado no ecrã;

Esta norma apresenta também algumas exigências e recomendações para os ícones. Por exemplo, uma das exigências é que a interacção com o ícone não pode destruir dados do utilizador sem permissão do mesmo.

Norma ISO/IEC 18019

Quanto à ISO/IEC 18019 (2004) [42], é uma norma que define os métodos para a preparação e construção de documentação para utilizadores de aplicações de *software*. Esta norma explica como:

- Determinar qual a informação que o utilizador necessita;
- Deve ser apresentada essa informação;
- Preparar a informação;
- Torná-la disponível;

A norma não indica que apenas existe uma forma correcta de desenvolver a documentação. No entanto, apresenta recomendações. O responsável pela produção da documentação deve avaliar o que é melhor de acordo com o utilizador alvo.

Os tópicos principais desta norma são os seguintes:

- Formatos da documentação;
- Definição de objectivos;
- Análise e *design*;
- Planeamento;
- Desenvolvimento e revisão;

2.3.2 Boas práticas

Algumas das boas práticas no desenvolvimento de aplicações são geralmente comuns à maioria dos *guidelines* existentes [43], sendo aplicáveis a vários tipos de produtos (*desktops*, computadores portáteis, PDAs, *smartphones*, *tablets*). Contudo, existem boas práticas que têm maior foco nos dispositivos móveis [44], principalmente devido às características específicas deste tipo de dispositivos, como por exemplo, o modo de interacção entre o utilizador e o dispositivo. A interacção com um *desktop* é necessariamente diferente da interacção com um *smartphone*. No *desktop*, a interacção entre o utilizador e a aplicação é essencialmente feita através do teclado, rato/*touchpad*. Num *smartphone*, a interacção é geralmente realizada utilizando um ecrã táctil (apesar de também existir com teclado). Estas

diferenças têm implicações e é neste contexto que surgem algumas recomendações específicas ao desenvolvimento de aplicações móveis.

Neste sentido, são apresentadas em seguida algumas boas práticas genéricas que se aplicam a todas as interfaces, seguidas de outras específicas ao desenvolvimento de interfaces móveis.

Genéricas

- **Visibilidade do estado do sistema:** O sistema deve manter os utilizadores informados acerca do que se está a passar durante a execução da aplicação;
- **Ligação entre o sistema e o exterior:** O sistema deve "falar" a língua do utilizador, ou seja, deve utilizar palavras, frases e conceitos que o utilizador consiga perceber. A informação deve aparecer numa ordem natural e lógica;
- **Função de anular e de refazer:** É normal o utilizador se enganar na escolha da operação que pretende efectuar, pelo que deve ser possível anular essa acção quando tal acontece. Deve existir uma forma simples de anular e refazer acções;
- **Informação relevante:** As caixas de diálogo devem apenas conter informação importante, a informação irrelevante ou raramente necessária diminui a atenção do utilizador do que é realmente importante;
- **Mensagens de erro claras:** Estas mensagens devem ter uma linguagem que identifica de forma objectiva o problema, e apresentação de proposta de solução;
- **Ajuda e documentação:** Embora os utilizadores devam conseguir utilizar o sistema sem necessidade obrigatória de documentação, isso pode ser necessário. Essa informação deve ser de fácil acesso e estar bem organizada;
- **Prevenção de situações anómalas:** Deve ser eliminada a origem deste tipo de problemas e tentar evitar a sua ocorrência. No caso de existir o risco de ocorrência de um problema, o utilizador deve ser previamente avisado;
- **Escolha das normas adequadas:** De forma a garantir consistência da aplicação com o restante sistema, devem ser escolhidas as normas adequadas para o desenvolvimento da aplicação;

Móveis

- **Inicialização rápida da aplicação:** O cenário comum para a utilização de aplicações é quando o utilizador inicia a aplicação, efectua algumas acções e fecha a aplicação passado pouco tempo depois de a ter executado. Por isso, é conveniente que o tempo de inicialização seja reduzido, principalmente se for esperado que a aplicação seja executada repetidamente;

- **Dimensionamento adequado dos controlos do UI[12]:** Sendo o toque no ecrã a forma como o utilizador interage com a aplicação, existe uma maior probabilidade de tocar no sítio errado da interface. Assim, os botões devem ter dimensões suficientemente grandes para minimizar este tipo de erros;
- ***Feedback* visual aquando da interacção com o UI:** Quando o utilizador toca num botão (ou noutro tipo de controlo) deve receber um *feedback* visual que indica que o botão foi efectivamente pressionado. Por exemplo, através da mudança de cor do controlo enquanto este se encontra pressionado;
- **Execução das acções desejadas quando o controlo for libertado:** Deste modo, o utilizador pode mover o dedo para outro local se tiver pressionado o botão errado e a função associada àquele botão;
- **Cada *screen*[13] com a sua funcionalidade:** Devido ao tamanho do ecrã ser reduzido em certos casos, em vez de tentar colocar muitas funcionalidades e controlos no mesmo *screen*, é preferível dividir as funcionalidades e criar *screens* para cada uma delas, fazendo depois a devida navegação entre elas. No caso de uma funcionalidade necessitar de mais espaço, o recurso ao *scrolling* é uma boa solução;
- ***Design* atractivo:** Em aplicações móveis um factor muito importante é o aspecto visual, por exemplo, ao nível da conjugação de cores, dos ícones, disposição dos vários elementos, pelo que o *design* deve ser o mais atractivo possível;
- **Controlos do UI intuitivos:** Alterar drasticamente o aspecto dos controlos fornecidos por defeito pela plataforma não é recomendável, pelo que quando se procede a este tipo de customização deve-se ter a certeza que continuam a ser intuitivos;

2.4 Projectos *Open Source*

O conceito de *Open Source* está relacionado com o desenvolvimento de *software*. Este método de desenvolvimento permite a quem desenvolve produtos de *software* os possa partilhar, com estes produtos a poderem ser melhorados e redistribuídos por outros, tudo isto sujeito a regras bem definidas. O modelo de desenvolvimento de *software open source* visa oferecer às entidades que desenvolvem *software* melhor qualidade, mais flexibilidade, custos mais reduzidos e maior fiabilidade [45].

Neste âmbito, surge a organização sem fins lucrativos Open Source Iniciative (OSI) [46] que tem como objectivo regulamentar o sector do *software open source*, evidenciar os benefícios do uso deste tipo de *software* e construir ligações entre os diferentes constituintes da comunidade *open source*. Esta organização apresenta uma definição de *open source* que

[12] Controlos do UI – os botões, por exemplo.
[13] UI apresentado num dado instante.

não significa apenas ter acesso ao código fonte. Resumidamente, a distribuição segundo estes termos deve seguir os seguintes princípios:

- **Redistribuição livre:** A licença não deve restringir ninguém de vender ou oferecer o *software*, ou exigir o pagamento de qualquer taxa (*Royalty*) para se poder efectuar a redistribuição de uma versão modificada do *software* original.
- **Código fonte:** O programa deve incluir o código fonte e deve permitir a distribuição desse mesmo código assim como da versão compilada. No caso de o *software* não incluir o código fonte, o método de obtenção do mesmo, sem encargos, deve ser explícito e de fácil acesso.
- **Produtos derivados:** A licença deve permitir modificações e produtos derivados e deve permitir que estes sejam distribuídos nos mesmos termos da licença original do *software*.
- **Integridade do código fonte dos autores:** A licença pode impor restrições no sentido de ser possível distinguir entre o código fonte original e o código fonte de versões modificadas. Por exemplo, a licença pode requerer que versões modificadas utilizem um nome diferente do *software* original.
- **Não discriminação de pessoas ou grupos:** A licença não pode discriminar qualquer pessoa ou grupo de pessoas.
- **Não discriminação da área de utilização:** A licença não pode limitar o uso do produto numa área de aplicação específica.
- **Distribuição da licença:** A licença deve poder ser aplicada a qualquer redistribuição, sem necessidade de criação de uma licença adicional por parte da entidade responsável pela nova distribuição.
- **Licença não deve ser específica de um produto:** Os direitos associados ao produto não devem ser dependentes de uma distribuição em particular.
- **Licença não deve restringir outro *software*:** A licença não deve impor restrições a outro *software* que seja distribuído como *software* licenciado. Por exemplo, a licença não pode impor que todos os outros programas que sejam distribuídos no mesmo meio sejam *software open source*.
- **Licença deve ser tecnologicamente neutra:** A licença não pode impor nenhuma tecnologia específica para o uso do *software*.

2.4.1 Licenças *open source*: perspectiva comercial

Sendo a OSI uma organização de normalização de *software open source*, é possível encontrar no seu *website* uma lista de licenças que cumprem a definição de *open source* apresentada anteriormente, apesar de cada uma apresentar condições contratuais diferentes.

Estas licenças foram aprovadas pela OSI através de um processo de revisão de licenças. De seguida são apresentadas algumas das licenças mais populares e que são amplamente utilizadas no âmbito do *software open source*.

Apache License, version 2.0 (Apache-2.0)

Esta versão, que data de 2004 [47], concede alguns direitos sob determinadas condições a quem recorre a esta licença para licenciar *software*. Numa perspectiva de utilização desta licença na comercialização de um produto (*software*), é apresentado de seguida, de forma resumida, os principais termos desta licença:

Tabela 5 - Alguns termos presentes nas licenças Apache-2.0

Permite:	Fazer o download e utilizar o *software* Apache[14], tanto a nível pessoal, empresarial ou por motivos comerciais;
	Utilização do *software* Apache no desenvolvimento de novas distribuições ou pacotes;
Proíbe:	A redistribuição de qualquer fragmento do *software* original sem a devida atribuição;
	Utilização de qualquer marca pertencente à The Apache Sotware Foundation que possa afirmar ou sugerir que esta fundação apoia a distribuição desenvolvida;
Requer:	Inclusão de uma cópia da licença Apache em qualquer redistribuição desenvolvida em que seja incluído *software* Apache;
	Fornecer atribuição clara ao The Apache Software Foundation em qualquer distribuição que inclua *software* Apache;

Esta licença é compatível com licenças GPLv3 (introduzida mais abaixo), o que significa que *software* Apache-2.0 pode ser incluído em projectos com licenças do tipo GPLv3. No entanto, *software* GPLv3 não pode ser incluído em projectos Apache. Existem vários projectos que recorrem a esta licença. Por exemplo, o Apache HTTP Server ou o Android.

GNU General Public License, version 2.0 (GPLv2)

Esta licença, publicada em 1991 [48], é também muito utilizada pela comunidade *open source* para proteger o *software* desenvolvido. Esta licença não exige que sejam tornadas públicas as versões modificadas do *software* abrangido pela mesma, ou seja, é possível desenvolver novas versões e utilizá-las, tanto a nível particular como a nível empresarial, sem ser necessário expor as modificações efectuadas, desde estas novas versões não sejam comercializadas. No entanto, se a versão modificada for tornada pública de alguma forma, a GPL requer que o código fonte dessa versão seja disponibilizado aos utilizadores desse programa, sob a licença GPL. A GPL permite a comercialização das versões modificadas mas com a imposição de que deve estar também sob a licença GPL, o que significa que é obrigatório que os utilizadores possam ter acesso (ou lhes seja fornecido) o código fonte do

[14] *Software* Apache – Software sob a licença Apache.

software. Não é permitida a publicação de versões modificadas apenas na forma de ficheiros binários/executáveis. Isto implica que ao se utilizar a licença GPL para licenciar determinado *software*, permite-se que este seja modificado e redistribuído nas mesmas condições do *software* original.

GNU General Public License, version 3.0 (GPLv3)

A GPLv3, publicada em 2007 [49], é uma versão mais completa da versão GPLv2, e é uma licença do tipo Copyleft (tal como a GPLv2), isto força a que quem tira partido de *software open source* ofereça os seus desenvolvimentos à comunidade *open source*. Esta versão contém termos que não se encontravam ou que não eram suficientemente abrangidos pela versão anterior, como por exemplo, ao nível da compatibilidade com outras licenças, das patentes do *software* e dos métodos de "tivoization[15]" [50]. Alguns exemplos de projectos que utilizam as licenças GPL são: Dynebolic, gNewSense, Musix ou o Ututo. Todos eles são distribuições GNU/Linux, mas estas licenças podem ser aplicadas em outros tipos de projectos/*software*.

Existem outras licenças *open source* tais como a Mozilla Public License version 1.1 (MPL-1.1), a Common Development and Distribution License (CDDL) e a Berkeley Software Distribution (BSD). De modo a reunir de forma compacta as características de algumas licenças *open source*, foi elaborada a Tabela 6 (adaptada de [51]):

Tabela 6 - Algumas características relativas a licenças *open source*

	Apache-2.0	GPL	MPL-1.1	CDDL	BSD
É obrigatório incluir código fonte?	Não	Sim	Sim	Sim	Não
É obrigatório publicar modificações?	Não	Sim	Sim	Não	Não
É obrigatório os trabalhos derivados continuarem a ser *open source*?	Não	Sim	Não	Não	Não
Possibilidade de utilização de outras licenças para trabalhos derivados?	Sim	Sim	Não	Sim	Sim
Produtos derivados podem ser comercializáveis?	Sim	Sim	Sim	Sim	Sim

Deve-se ter em consideração que a descrição e tradução de documentos legais pode originar erros de interpretação, pelo que para se ter uma descrição exacta e correcta da informação é altamente recomendável a leitura e compreensão dos documentos oficiais antes de se decidir pela aplicação de qualquer uma das licenças.

[15] Métodos de restrição de desenvolvimento de *software* livre em que quando o *hardware* detecta versões modificadas, desliga o sistema.

Pode-se concluir que as empresas que procuram desenvolver produtos baseados em *software open source*, numa perspectiva de poder comercializar esses mesmos produtos, necessitam de ter em conta o tipo de licença do *software* original para não terem surpresas. Por exemplo, como referido, nas licenças GPL é exigido que o código fonte dos produtos derivados seja disponibilizado, permitindo que esse mesmo produto possa ser modificado e redistribuído por outras entidades. Isto pode dificultar a utilização deste tipo de *software* por parte das empresas, visto que os produtos desenvolvidos passam a poder ser utilizados/modificados livremente pela comunidade. Contudo, existem licenças que fornecem mais garantias às empresas na óptica da comercialização, como é o caso da licença Apache-2.0. Esta licença é propícia a ser utilizada em *software* cujo objectivo seja a comercialização, pois permite a redistribuição na forma de *software* proprietário, isto é, com restrições ao nível da utilização, modificação, cópia e distribuição, salvaguardando deste modo os interesses das empresas.

2.4.2 O caso particular do Android

O Android é um projecto que tem evoluído bastante desde o seu aparecimento em 2003, sobretudo após a Google ter adquirido a Android Inc. em 2005 e esta ter integrado a formação da Open Handset Alliance em 2007. Em 2008 torna-se um projecto *open source*.

O projecto Android recorre a algumas das licenças *open source* aprovadas pela OSI para o desenvolvimento de *software* nesta plataforma. A sua licença de eleição para a maioria do *software* desenvolvido é a Apache-2.0, embora também possam ser escolhidas licenças similares como a BSD. A opção por este tipo de licenças deve-se ao facto do Android não acreditar que as entidades (empresas, por exemplo) invistam nesta plataforma se não poderem retirar proveitos dos avanços efectuados no desenvolvimento de *software*, visto que com outro tipo de licenças como as GPL, as entidades são obrigadas a tornar o código fonte acessível e sujeito a modificações e redistribuição. Contudo, o Android encoraja o desenvolvimento de *software* que seja livre e modificável. Deste modo, a licença Apache-2.0 permite às empresas adoptar esta plataforma e desenvolver aplicações para as camadas superiores sem ser obrigatório expor os trabalhos desenvolvidos (código fonte).

3. Plataforma B-Live

Esta dissertação enquadra-se no projecto *Living Usability Lab for Next Generation Networks* (LUL), em particular, na contribuição da empresa Micro I/O para o desenvolvimento de uma rede de comunicação sem fios de suporte a vários sensores e actuadores. Nesta secção apresenta-se o projecto LUL, bem como as arquitecturas de *hardware* e *software* do sistema B-Live Wireless.

3.1 Living Usability Lab

O projecto LUL tem como objectivo desenvolver serviços e tecnologias que permitam tornar a população idosa mais saudável, mais activa e mais produtiva no seu lar, tendo especial atenção a especificações de usabilidade, dado variarem com a faixa etária dos utilizadores alvo. Contudo, apesar do projecto LUL estar direccionado especialmente para os idosos, as tecnologias e serviços desenvolvidos também poderão ter impacto na população em geral, assim como em cidadãos com necessidades especiais de carácter permanente ou temporário. O projecto tira partido das redes de nova geração e da computação distribuída de modo a atingir os objectivos propostos. A Figura 7 apresenta uma visão geral do projecto.

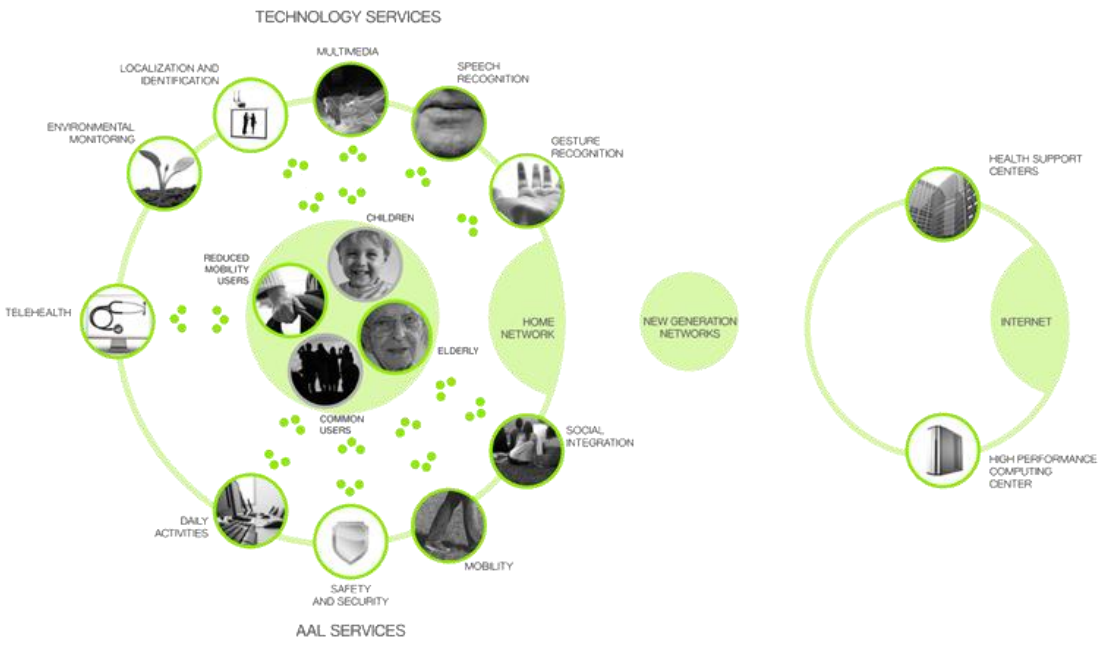


Figura 7 – Visão geral sobre o projecto LUL

Fonte: *Living Usability Lab* [5]

Como se pode observar, os serviços e tecnologias têm o intuito de poderem ser utilizados por todas a faixas etárias, não só por pessoas com saúde, mas também por pessoas com dificuldades funcionais. Reconhecimento de voz e gestos, localização e identificação, telemedicina, mobilidade, segurança social, segurança e monitorização do ambiente, são alguns dos serviços e tecnologias que o projecto LUL se propõem a desenvolver, através da integração de redes de nova geração, para aceder a serviços de suporte médico e serviços de computação distribuída.

Para testar as tecnologias e os serviços desenvolvidos, o projecto visa construir laboratórios vivos nos quais são implementados ambientes e cenários de utilização real, onde existe a possibilidade de os utilizadores finais (cidadãos seniores, pessoas com necessidades especiais, cidadãos em geral) poderem experimentar estas tecnologias e serviços.

Neste sentido, a Micro I/O está a preparar um laboratório vivo a ser instalado na Escola Superior de Saúde da Universidade de Aveiro (ESSUA) que será composto por vários sensores e actuadores sem fios, *hardware* para a comunicação entre os diversos dispositivos e um servidor que suportará as aplicações desenvolvidas. O conjunto destes elementos é designado por sistema B-Live Wireless. Antes da instalação no laboratório, foi criado um pré-piloto que foi montado no departamento de I&D da Micro I/O de forma a serem realizados testes preliminares para verificar o comportamento e desempenho do sistema.

Devido a esta divisão entre o piloto da ESSUA e o pré-piloto da Micro I/O, serão desenvolvidas duas versões da aplicação móvel: uma para o pré-piloto e outra para o piloto. Estas duas versões terão algumas diferenças em termos do aspecto visual, mas a principal diferença será ao nível da tecnologia utilizada para interagir com o sistema B-Live. Para o pré-piloto a aplicação interage com o servidor através de um socket TCP, usando mensagens proprietárias, enquanto que para o piloto, a interacção é suportada em *Web Services*. Estas duas implementações serão descritas com maior pormenor adiante.

3.2 Sistema B-Live Wireless

O B-Live Wireless é um sistema desenvolvido pela Micro I/O, que tem como objectivo o desenvolvimento de tecnologia para domótica habitacional baseada em comunicações sem fios [52]. Os principais elementos que compõem o sistema são uma rede local de sensores e actuadores (WSAN) e um servidor. A Figura 8 representa a estrutura de rede que suporta o sistema. A WSAN é baseada no *standard* IEEE 802.15.4. O servidor é o responsável por processar e/ou armazenar toda a informação proveniente da WSAN, assim como disponibilizar a informação para aplicações de controlo. A sua arquitectura está descrita no subcapítulo 3.2.2 desta secção. A comunicação entre a WSAN e o servidor é efectuada através de uma ligação Ethernet (LAN[16]) ou por Wi-Fi (WLAN[17]). A interligação entre as tecnologias IEEE 802.15.4 e Ethernet/Wi-Fi é realizada com recurso a um *gateway*. Na imagem, a rede de sensores e actuadores é representada por uma WPAN[18].

[16] LAN – *Local Area Network,* rede de dados cablada geralmente a funcionar de acordo com o protocolo IEEE 802.3 (Ethernet).
[17] WLAN – Wireless LAN, redes de dados sem fios geralmente a funcionar de acordo com o protocolo IEEE 802.11 (Wi-Fi).
[18] WPAN – Wireless Personal Area Network, rede de dispositivos de curto alcance.

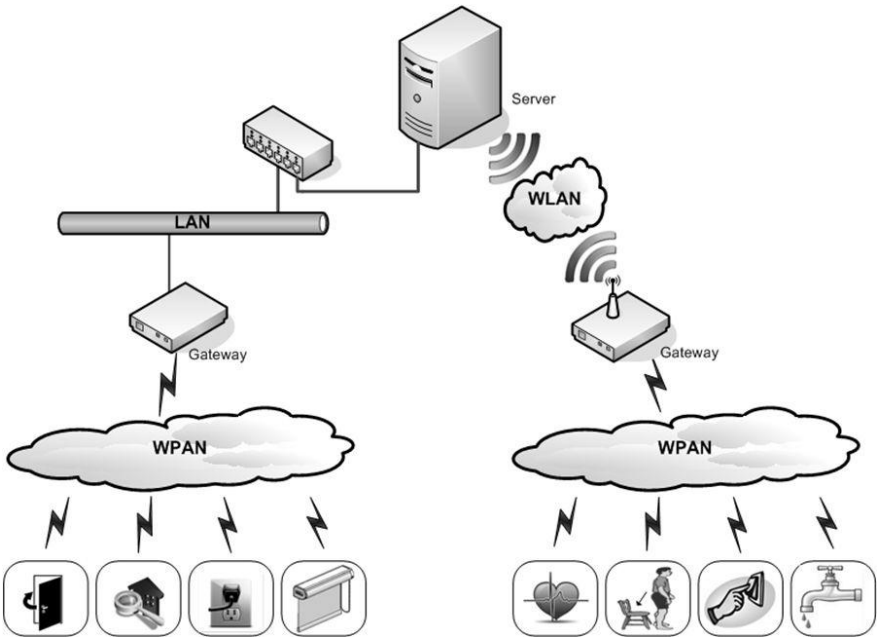


Figura 8 – Arquitectura da rede B-Live Wireless

Fonte: *B-Live Wireless: A Real-Time Protocol For Home Automation* [52]

De seguida são apresentadas as arquitecturas de *hardware* e de *software* do sistema. Devido às diferenças entre as arquitecturas do pré-piloto instalado na Micro I/O e do piloto da ESSUA, as mesmas irão ser abordadas separadamente.

3.2.1 Arquitectura de *Hardware*

Pré-piloto (Micro I/O)

A arquitectura de *hardware* do sistema instalado no departamento de I&D da Micro I/O está representado na Figura 9. Como se pode observar, o sistema é composto por um conjunto de sensores e actuadores que possibilitam a monitorização e o controlo de diversos elementos. A lista de *hardware* presente no pré-piloto é o seguinte (mapeamento na Figura 9) [53]:

1) Servidor: computador responsável pelo processamento e armazenamento da informação proveniente da WSAN e por fornecer informação a aplicações de controlo, através de *sockets* TCP.

2) *Gateway*: dispositivo que realiza a ligação entre a WSAN e o servidor B-Live usando Ethernet.

3), **12)**, **17)** e **18)** Sensor magnético: sensor que monitoriza o estado (aberto/fechado) de portas (de entrada e dos armários) e da janela.

4), **7)**, **8)** e **11)** Actuador com relé: dispositivo que permite controlar remotamente a alimentação a cargas AC. É utilizado em lâmpadas e tomadas, permitindo ligar e desligar as mesmas.

5) Sensor de ultra-sons: comunica periodicamente a distância a que o objecto mais próximo se encontra.

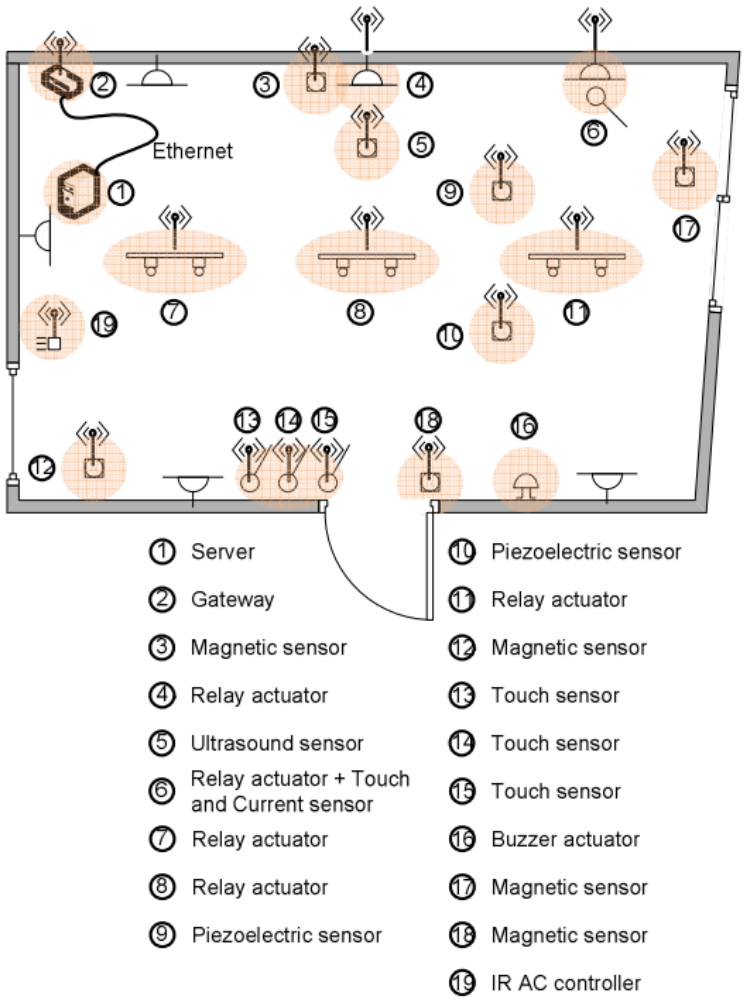


Figura 9 – Arquitectura de *hardware* do pré-piloto do sistema B-Live Wireless

Fonte: Pré-piloto B-Live Wireless [53]

6) Actuador de relé, comutador de toque e sensor de corrente: dispositivo integrado que monitoriza o consumo da tomada e tem a possibilidade de ser ligada/desligada através de toque ou através de uma aplicação remota.

9) e **10)** Sensor piezoeléctrico: detecta variações bruscas de pressão. Estes sensores são utilizados para detectar pancadas na secretária.

13), **14)** e **15)** Sensor de toque: detecta a proximidade física de uma mão. São utilizados como interruptores para as lâmpadas da sala.

16) Actuador *buzzer*: permite emitir diversos sons com uma configuração específica. É utilizado por exemplo, para emitir sons quando a porta da sala se encontra aberta.

19) Controlador de infra-vermelhos: permite actuar sobre equipamentos controlados por infra-vermelhos, como por exemplo, o ar condicionado.

Piloto (ESSUA)

A Figura 10 documenta a arquitectura de *hardware* do piloto do sistema B-Live Wireless a instalar na ESSUA [54]. O sistema é composto por um conjunto alargado de sensores e actuadores que permitirão monitorizar e controlar diversos elementos existentes no laboratório.

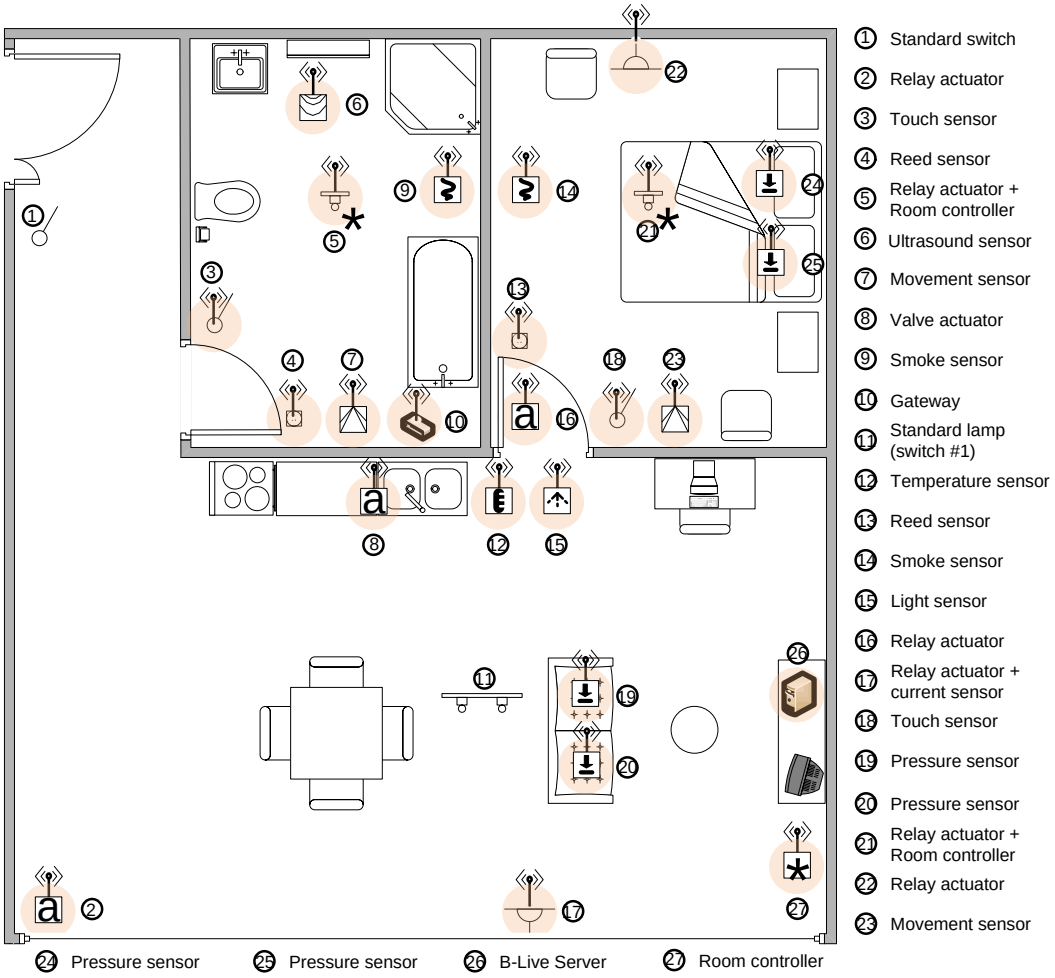


Figura 10 - Arquitectura de *hardware* do piloto do sistema B-Live Wireless

Fonte: Piloto B-Live Wireless [54]

Recorrendo à Figura 10, é possível mapear o seguinte *hardware* que constitui o sistema:

1) Comutador simples: comutador de iluminação para a lâmpada da sala **11)**. Estes dois dispositivos (1 e 11) não integram o sistema B-Live Wireless.

2) Actuador com duplo relé: permite controlar remotamente o estore da janela do laboratório.

3) e **18)** Sensor de toque: detecta a proximidade de uma mão. Cada sensor está associado a um ou mais actuadores de lâmpada, possibilitando a sua comutação.

4) e **13)** Sensor magnético: monitoriza o estado (aberto ou fechado) das portas.

5) e **21)** Actuador com relé: permite controlar remotamente a alimentação de cargas AC, neste caso lâmpadas. Este dispositivo opera também como controladores de divisão, ou seja, é responsável por comunicar ao *gateway* todos os eventos detectados na divisão que está associado.

6) Sensor de ultra-sons: comunica periodicamente a distância a que o objecto mais próximo se encontra. Neste caso é usado para detectar a presença de pessoas junto ao espelho.

7) e **23)** Sensor de movimento: detecta a ocorrência de movimento numa divisão usando um sensor PIR (*Passive Infrared Sensor*). Utilizado para detectar movimentos nas divisões do quarto e da casa de banho.

8) Actuador com relé e electroválvula: comuta o estado de uma electroválvula remotamente. É utilizado na torneira da água da banca.

9) e **14)** Sensor de fumo: detecta a existência de fumo numa divisão utilizando tecnologia de infra-vermelhos.

10) *Gateway*: faz a interligação entre a rede *wireless* de sensores e actuadores (IEEE 802.15.4) e o servidor B-Live usando Ethernet.

12) Sensor de temperatura: Comunica periodicamente a temperatura da divisão sala.

15) Sensor de luminosidade: Comunica periodicamente a luminosidade da divisão sala.

16) Actuador com relé: controla o motor associado à porta de forma a realizar a sua abertura ou fecho.

17) Actuador com relé e sensor de corrente: dispositivo integrado que comuta o estado da tomada e monitoriza o consumo da mesma após esta ter sido ligada.

19), **20)**, **24)** e **25)** Sensor piezoresistivo: detecta variações de pressão que ocorrem quando alguém se deita na cama ou senta no sofá.

22) Actuador com relé: comuta remotamente a alimentação a cargas AC, neste caso, o equipamento que esteja ligado à tomada.

26) Servidor: computador responsável pelo armazenamento da informação de estado da rede de sensores e actuadores bem como de solicitar a informação em resposta a pedidos provenientes de serviços *Web*.

27) Controlador de divisão: responsável por comunicar ao *gateway* todos os eventos detectados na divisão em que está instalado.

3.2.2 Arquitectura de *Software*

Pré-piloto (Micro I/O)

Na Figura 11 está representada a arquitectura de *software* do servidor que suporta o sistema B-Live Wireless [53]. Este servidor é essencialmente constituído por uma base de dados MySQL onde é armazenada toda a informação relevante do sistema (estado dos dispositivos, informação sobre os utilizadores, descrição do local, etc.), e uma aplicação

servidora, desenvolvida em linguagem Java. Esta aplicação é responsável pelo processamento da informação, fazendo a interface entre a base de dados e o exterior, tanto para a WSAN como para as aplicações de controlo. A interacção entre a aplicação servidora e a base de dados é feita via JDBC (*Java DataBase Connectivity*: API em Java para manipular bases de dados SQL), enquanto a comunicação com a WSAN e as aplicações de controlo é realizada utilizando um protocolo proprietário da Micro I/O que opera sobre *sockets* TCP/IP.

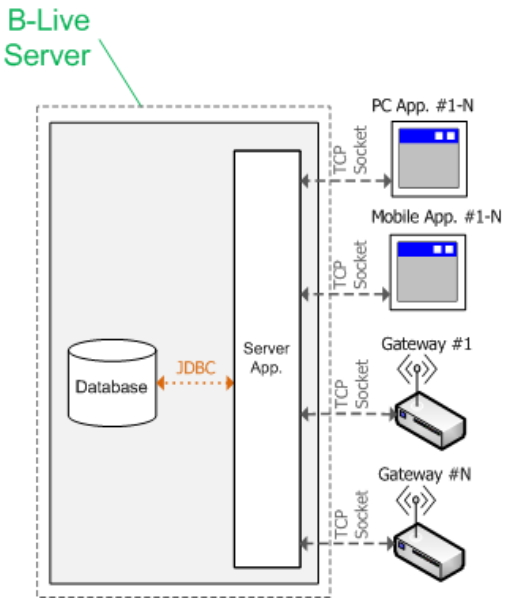


Figura 11 - Arquitectura de *software* do pré-piloto do sistema B-Live Wireless

Fonte: Pré-piloto B-Live Wireless [53]

Piloto (ESSUA)

A Figura 12 apresenta a arquitectura de software do servidor do sistema B-Live Wireless a instalar na ESSUA [54]. O servidor pode utilizar qualquer sistema operativo, e realiza a interface com o exterior através de *Web Services*.

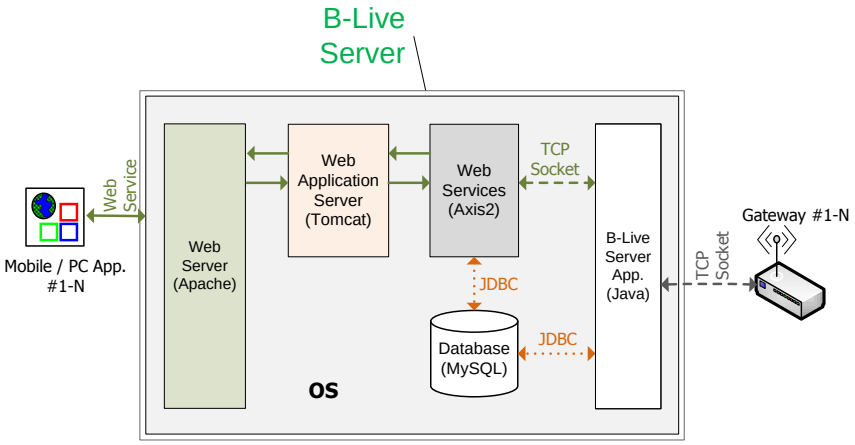


Figura 12 - Arquitectura de *software* do piloto do sistema B-Live Wireless

Fonte: Piloto B-Live Wireless [54]

Um servidor Apache é utilizado para suportar os serviços. Os serviços Web serão suportados na *stack* Axis2 [55] e no servidor de aplicações Web Tomcat [56]. A informação relativa ao sistema B-Live Wireless instalado encontra-se armazenada numa base de dados MySQL e os serviços *Web* permitem o acesso remoto a esta informação. O acesso à base de dados é realizado através da API JDBC. Adicionalmente, os *Web Services* poderão também solicitar o envio de comandos para a rede de sensores/actuadores. Além de ser responsável por receber informação da rede de sensores/actuadores e proceder à sua inserção na base de dados, a aplicação servidora (desenvolvida em Java) está também encarregue de encaminhar os pedidos de alteração de estado provenientes dos serviços Web para a rede de sensores/actuadores. A comunicação entre a aplicação servidora e o *gateway* é realizado de utilizando um protocolo proprietário da Micro I/O que opera sobre *sockets* TCP/IP.

4. A aplicação B-Live Home

O acesso à informação em qualquer lugar e em qualquer instante é cada vez mais importante na vida das novas gerações. É expectável um crescimento significativo quer na utilização de dispositivos móveis quer no acesso à informação através dos mesmos dispositivos. Nesta sequência considerou-se que seria muito importante a interacção com o sistema B-Live Wireless instalado numa habitação a partir de um dispositivo móvel, local ou remotamente. Um cenário típico da utilidade desta solução seria, por exemplo, um utilizador sair de casa e esquecer-se do aquecedor ligado. Sem a possibilidade de remotamente actuar sobre o estado do aquecedor, o mesmo poderia ficar ligado até que alguém regressasse a casa e o desligasse fisicamente. Neste cenário consumir-se-iam recursos desnecessariamente. Com uma solução com suporte para dispositivos móveis, o utilizador poderia verificar (ou ser avisado) que o aquecedor ficou ligado e solicitar que este fosse desligado, economizando energia.

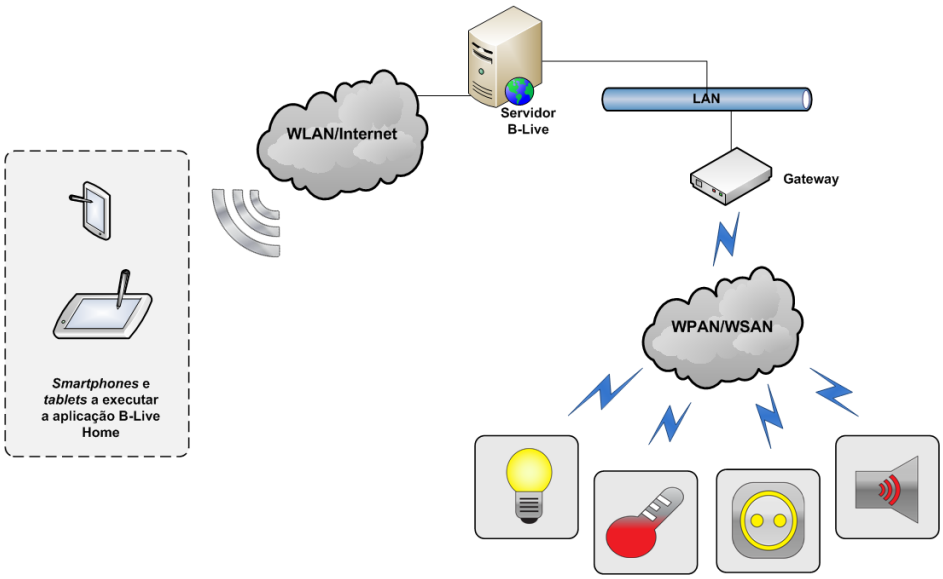


Figura 13 - Enquadramento da aplicação B-Live Home no sistema B-Live Wireless

Entendeu-se ser útil o desenvolvimento de uma aplicação para terminais móveis que permitisse interagir com o sistema B-Live Wireless. Em resultado, foi proposta a aplicação B-Live Home destinada a estes dispositivos. Como se pode observar na Figura 13, esta aplicação interage como o sistema comunicando com o servidor B-Live. A interacção com o sistema é, contudo, realizada de forma diferente nos dois pilotos:

- No pré-piloto instalado na Micro I/O a comunicação é realizada através de um *socket* TCP;

- No piloto a instalar na ESSUA a comunicação é realizada através de *Web Services* disponibilizados pelo servidor B-Live.

A ligação entre a aplicação e o servidor pode ser efectuada tanto através de uma rede local (WLAN), como através da Internet[19].

A plataforma seleccionada para o desenvolvimento da aplicação é Android, como se verá adiante. Esta plataforma oferece várias vantagens para o desenvolvimento de uma aplicação deste tipo, destacando-se:

- Variedade de dispositivos em que esta plataforma é executada;
- Quota de mercado muito significativa;
- Elevada flexibilidade;
- Baixo custo de desenvolvimento;
- Solução *open source*;

O facto de ser *open source*, libertada de acordo com os termos da licença Apache 2.0, é muito importante pois permite que a libertação de trabalhos derivados seja efectuada de acordo com outro tipo de licença, como por exemplo, uma licença proprietária.

O Android apresenta diversas versões da plataforma, pelo que é necessário ter em consideração as questões de compatibilidade da aplicação entre versões. Baseado no número de dispositivos que acederam ao Google Play, a Google disponibiliza as percentagens de dispositivos com cada versão. A Figura 14 representa essa distribuição (valores retirados do site no dia 27-5-2012). Como se pode observar, as versões mais usadas são as do Android 2.3.3 até 2.3.7. Sendo assim, a versão escolhida para o desenvolvimento da aplicação é o Android 2.3.3. Isto significa que não é possível executar a aplicação em versões inferiores ao 2.3.3 pois como as versões anteriores da plataforma não incluem as novas APIs disponibilizadas para esta versão, o sistema operativo é incapaz de executar a aplicação. Contudo, deve ser possível executar a aplicação em versões superiores pois praticamente todas as mudanças nas APIs são aditivas, excepto em casos isolados em que a aplicação utiliza partes da API que foram removidas em versões posteriores.

[19]No âmbito dos dois pilotos e numa fase inicial, a comunicação apenas pode ser realizada numa WLAN.

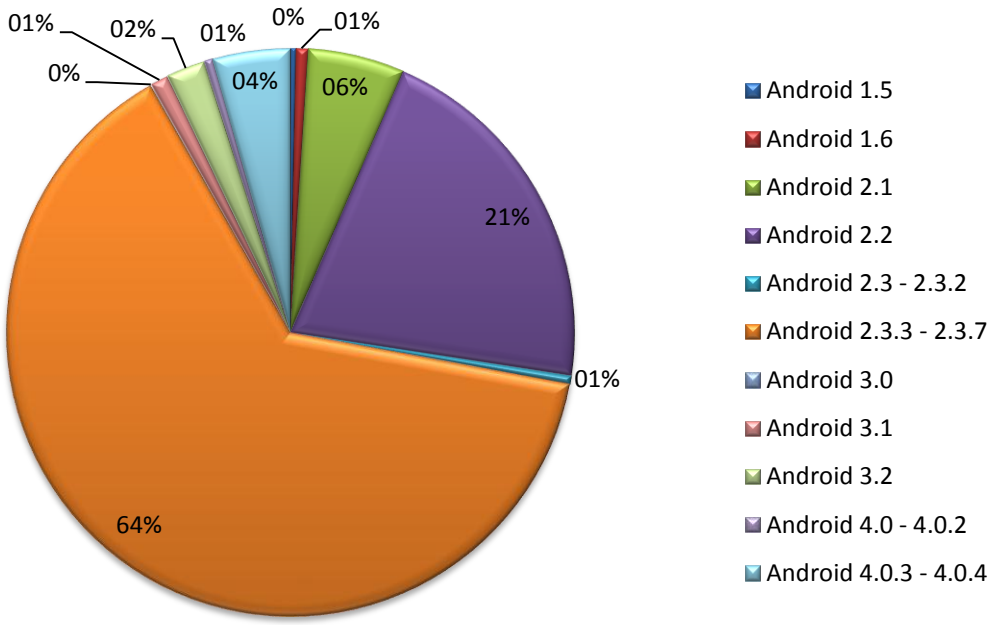


Figura 14 - Distribuição das versões da plataforma Android

Fonte: *Platform Versions [57]*

4.1 Requisitos

A aplicação B-Live Home deve satisfazer um conjunto de requisitos (funcionais e não funcionais) que permitam considerar aspectos como a usabilidade, a segurança e o desempenho. Os requisitos a implementar na aplicação são os seguintes:

- Visualização de informação correspondente aos diversos dispositivos (sensores e actuadores) presentes no sistema;
- Capacidade para actuar sobre os dispositivos possíveis de ser controlados (actuadores);
- Visualização dos dados pessoais do utilizador que está a interagir com o sistema através da aplicação;
- Autenticação no sistema;
- Possibilidade de agrupar os dispositivos do sistema de acordo com localização ou por tipo de dispositivo;
- Possibilidade de ordenar os dispositivos por ordem alfabética ou pela actualização mais recente;
- Suporte para diversos tipos de terminais móveis e diferentes características de ecrã;

- Suporte para mudanças de orientação do *display* em *runtime*.
- Manutenção do UI rápido e responsivo;

Os dois primeiros requisitos apresentados constituem a principais funcionalidades da aplicação. Estas funcionalidades permitem que o utilizador da aplicação possa visualizar o estado dos dispositivos e controlar os elementos que se encontram associados aos actuadores. As informações passíveis de serem visualizadas são, por exemplo, o estado das portas e janelas (aberta/fechada), das lâmpadas e tomadas (ligada/desligada), detecção de movimento (movimento detectado/a aguardar ocorrência de movimento), valor do consumo de corrente, valor da temperatura. A outra funcionalidade está relacionada com a possibilidade de o utilizador poder visualizar a informação que lhe está associada na base de dados, tais como o nome, idade, nacionalidade, número de telefone e *email*.

Por questões de segurança, a informação apenas é acessível a quem está devidamente autorizado, isto é, a quem está registado na base de dados do sistema B-Live. Deste modo, a aplicação deve possuir um processo de autenticação que preceda a interacção com o sistema, recorrendo a um nome de utilizador e uma palavra-chave.

O B-Live Wireless é um sistema hierárquico, permitindo que se possa caracterizar um dispositivo ao nível da sua localização na habitação e ao nível do tipo de dispositivo. Assim, a aplicação deve possibilitar a filtragem dos dispositivos a apresentar em diversas categorias: sensores, actuadores, por divisão (por exemplo, sala ou quarto) e por tipo (por exemplo, lâmpada, tomada, janela). Também deve ser possível ordenar a apresentação dos dispositivos por ordem alfabética e por instante temporal da última alteração.

Sendo o Android uma plataforma suportada por vários tipos de dispositivos, com características diversas, é crucial que a aplicação seja executada correctamente no maior espectro possível de dispositivos. Um dos aspectos diferenciadores entre dispositivos é o tipo de ecrã e as suas características, tais como o tamanho, a resolução e a densidade (quantidade de pixéis numa área física do ecrã). Estas características devem ser tidas em consideração no desenvolvimento da aplicação para que o UI seja apresentado devidamente tanto em ecrãs de dimensões mais reduzidas (por exemplo, *smartphones*) como naqueles de dimensões superiores (por exemplo, *tablets*). Nos terminais móveis, a mudança de orientação do ecrã é frequente, pelo que estas situações devem ser precavidas de modo a optimizar a ocupação do espaço disponível. No sentido de testar a aplicação para dispositivos com diferentes características de ecrã, serão utilizados 3 dispositivos com as características apresentadas na Tabela 7 (as classificações encontram-se descritas no ponto 3 do Anexo 4).

Tabela 7 – Dispositivos móveis utilizados nos testes da aplicação

Dispositivo móvel	Tamanho	Densidade
Samsung Galaxy Gio[58]	*Normal* (3,2'')	*Medium*
Samsung Galaxy Note[59]	*Large* (5,3'')	*High*
Asus Eee Transformer Prime[60]	*Extra-large* (10,1'')	*High*

Por fim, é necessário ter em consideração que em relação aos dispositivos fixos, a maioria dos dispositivos móveis possui uma menor capacidade de processamento e de memória. Contudo, ainda assim, é necessário manter o UI rápido e eficiente em particular em cenários onde várias aplicações são executadas em simultâneo. Neste sentido, devem ser

usadas técnicas para garantir a responsividade da aplicação, nomeadamente recorrendo ao uso de *threads* (processos executados em paralelo).

4.2 Definição

A definição da aplicação é baseada nos requisitos apresentados no tópico anterior. A Figura 15 apresenta o diagrama de *use case* utilizando a linguagem de especificação e modelação UML (*Unified Modeling Language*), que pretende apresentar as principais funcionalidades da aplicação graficamente (na secção 4.3.2 são apresentados algumas características desta linguagem).

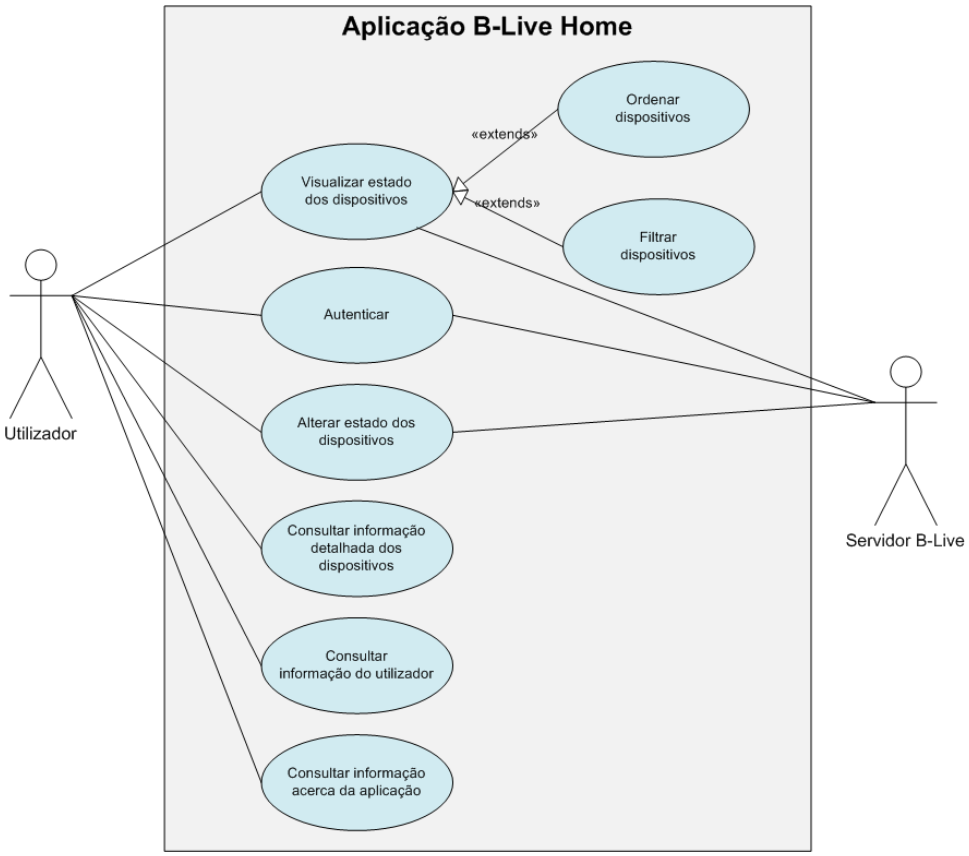


Figura 15 – Diagrama de *use case* da aplicação B-Live Home

Através da observação do diagrama é possível verificar a existência de um actor (Utilizador) que é a pessoa que utiliza a aplicação para tirar partido das funcionalidades da mesma. O outro actor representado no diagrama é o sistema B-Live Wireless, através do seu servidor. As funcionalidades estão representadas pelas elipses a azul que são denominadas de *use cases*. Quando as relações entre *use cases* são do tipo <<extends>> significa que um *use case* "herda" as características do outro e adiciona mais algumas. Neste caso, os *use cases* "Ordenar dispositivos" e "Filtrar dispositivos" têm o mesmo comportamento que o *use case*

"Visualizar estado dos dispositivos", sendo casos especiais (e opcionais) deste último. Quanto às relações entre o actor e os *use cases*, representam a capacidade desse actor interagir com o sistema de modo a recorrer a determinada funcionalidade.

Outro passo muito importante na definição de uma aplicação de *software* é a produção de um *mockup* da mesma, isto é, um modelo prévio que permite ilustrar de forma simples e por vezes não muito elaborada o aspecto visual da aplicação a desenvolver, bem como apresentar o modo como a aplicação responde aquando da interacção com o utilizador. Neste sentido, e tendo em consideração o diagrama de *use cases* apresentado anteriormente, foi elaborado um *mockup* da aplicação utilizando o *software* Balsamiq Mockups[20] [61] da Balsamiq Studios como o objectivo de ter uma percepção prévia do modo como a aplicação interage com o utilizador.

De seguida são apresentados os principais *layouts* definidos para a aplicação. Aquando da implementação, foram efectuadas algumas alterações relativamente ao que está representado neste *mockup* com o intuito de optimizar a ocupação do espaço, melhorando assim o UI da aplicação.

Na Figura 16 está o *layout* inicial da aplicação onde é apresentada uma barra de progresso como elemento principal e que significa que a aplicação está a inicializar. Este *layout* é apresentado durante alguns segundos e durante este tempo a aplicação pode, por exemplo, inicializar estruturas de dados necessárias, configurar parâmetros prévios ou até verificar se existe conexão ao servidor B-Live Wireless. Na Figura 17 é possível observar o *screen* utilizado para o processo de autenticação no sistema, em que o utilizador deve inserir o seu nome de utilizador e palavra-chave associados ao sistema B-Live e pressionar o botão de "*Login*". O botão "Limpar" serve para eliminar o que estiver nos campos anteriores. Existe também a possibilidade de relembrar os dados para que numa utilização posterior não seja necessário voltar a introduzi-los.

Figura 16 – *Mockup: Layout* inicial

Figura 17 - *Mockup*: *Layout* de *login*

[20] Versão *trial*: http://www.balsamiq.com/download.

Na Figura 18 está representado o *layout* que é apresentado quando a autenticação é efectuada com sucesso, com a listagem das possíveis funcionalidades da aplicação, embora apenas uma delas seja implementada inicialmente (funcionalidade de Estado).

Figura 18 – *Mockup: Layout* de apresentação das funcionalidades

Ao seleccionar-se a funcionalidade de Estado, surge um *layout* semelhante ao da Figura 19, no qual é possível observar a lista dos dispositivos existentes no sistema. Cada item da lista é composto pelo ícone que representa o dispositivo (deve reflectir o estado actual do dispositivo em causa), o nome e uma descrição do estado. Para além disto, no caso dos actuadores, deve também existir um controlo (por exemplo, botão) que permita alterar o estado do dispositivo. No canto superior direito do *layout* está a identificação do utilizador autenticado. Através do toque nesta identificação surge um menu em que é possível efectuar acções relacionadas com o utilizador como visualizar a sua informação associada ou terminar a sessão. Outro elemento que compõe o *layout* é o tipo de filtro aplicado na visualização dos dispositivos (na figura encontra-se por divisão). Quanto à Figura 20, mostra o menu de opções para este tipo de *layout*, no qual é possível voltar ao inicio da aplicação, actualizar a lista de dispositivos, alterar o tipo de filtro e escolher o tipo de ordenação na lista.

Figura 19 - *Mockup: Layout* da listagem dos dispositivos

Figura 20 - *Mockup: Layout* da apresentação do menu de opções

Quando é escolhido no menu da Figura 20 anterior a opção "Vista", ou seja, para escolher o tipo de filtro, são apresentadas as diversas opções de filtragem, como por exemplo, por "Tipo de dispositivo" ou "Divisão" em que estes se encontram. Este *layout* corresponde ao da Figura 21.

Figura 21 - *Mockup: Layout* para escolha do tipo de filtro a aplicar

Figura 22 - *Mockup: Layout* para um tipo de filtragem possível 1

Figura 23 - *Mockup: Layout* para definir a filtragem

No *layout* da Figura 21 é possível seleccionar um tipo de filtro. Ao escolher-se, por exemplo, “Tipo de dispositivo”, são apresentados apenas os dispositivos de um tipo (Figura 22). Quando se toca na zona em que está indicado o filtro definido é possível alterar o tipo de dispositivo, tal como está representado na Figura 23 e Figura 24.

Figura 24 - *Mockup: Layout para* um tipo de filtragem possível 2

Figura 25 - *Mockup: Layout* para filtragem por sensores

Figura 26 - *Mockup: Layout* para filtragem por actuadores

Por fim, é possível observar na Figura 25 e na Figura 26 a possibilidade de filtragem dos dispositivos por sensores e actuadores, com recurso às *tabs* que se encontram na parte superior do *layout*.

4.3 Implementação

Como já foi referido, o Android foi a plataforma seleccionada para o desenvolvimento da aplicação móvel devido às vantagens que apresenta em relação às outras plataformas existentes. No Anexo 3 encontra-se um guia passo a passo para a instalação e configuração do ambiente de desenvolvimento nesta plataforma.

A implementação desta aplicação pode ser divida em duas componentes principais:

- *Design*: onde se aborda tudo o que está relacionado com o aspecto visual da interface, desde as imagens, ícones, *layouts*, cores, formas e efeitos visuais.
- *Core*: relacionado com a parte funcional da aplicação propriamente dita, isto é, estruturas de dados, fluxos de navegação e de execução da aplicação e comunicação com o servidor.

Seguidamente, estas duas componentes são apresentadas de modo relativamente independente e quando apropriado, durante a apresentação de uma das componentes podem ser feitas referências à outra componente. Esta secção está fortemente relacionada com o Anexo 4 – Guias de referência e conceitos para desenvolvimento em Android. Como o próprio nome do anexo indica, este contém informação acerca dos guias de referência e dos conceitos disponibilizados pela Google para o desenvolvimento de aplicações em Android, pelo que

partes deste anexo serão invocadas sempre que necessário. No primeiro ponto do Anexo 4 é apresentada a estrutura típica de um projecto Android.

Outro dos aspectos que já foi referido é o facto de se ter desenvolvido duas aplicações para dois cenários diferentes: uma para o pré-piloto instalado na Micro I/O e outro para o piloto a instalar na ESSUA. Nesta secção é apresentado o desenvolvimento das duas aplicações, sendo evidenciado as diferenças entre as duas, tanto ao nível do *design* como do *core*, sempre que assim se justificar.

4.3.1 *Design*

O *design* básico da aplicação pode ser decomposto em três componentes: os ícones, as imagens e os *layouts*. A formulação destes componentes foi baseada nos *guidelines* fornecidos pela Google para a versão 2.3 da plataforma, no entanto, sempre que são lançadas novas versões, existem alterações a esses *guidelines*, que em alguns casos podem ser bastante significativas.

Um dos aspectos a ter em consideração no desenvolvimento do aspecto visual de uma aplicação é confirmar que estas características sejam apresentadas apropriadamente em vários tipos de ecrã, ou seja, em ecrãs com características diferentes, por exemplo, no que toca ao tamanho ou à densidade de pixéis numa área do ecrã. No ponto 3 do Anexo 4 está apresentada uma descrição desta situação, sendo apresentadas algumas técnicas para suportar os diferentes tipos de ecrã.

Para além da disposição dos elementos no UI, outra das características importantes são as cores envolvidas. Neste sentido, e recorrendo a ferramentas disponibilizadas *online*, tais como o Color Scheme Designer [62], procurou-se identificar um conjunto de cores com harmonia visual. Contudo, durante o desenvolvimento das aplicações, considerou-se conveniente a alteração do conjunto de cores para a aplicação do piloto (ESSUA) relativamente à aplicação do pré-piloto. A aplicação do pré-piloto contém tons predominantemente cor-de-laranja, em conjunto com verde e cinzento para o fundo. Na aplicação do piloto optou-se por tons predominantemente verde (tonalidade do ícone da Universidade de Aveiro, onde será instalado o piloto). Esta diferenciação é mais perceptível nos *layouts* apresentados adiante.

De seguida são apresentados os ícones, imagens e *layouts* produzidos para as duas aplicações.

Ícones

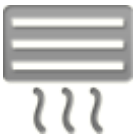
A aplicação é composta por vários ícones, que podem ser agrupados nas seguintes categorias:

- Ícones de estado;
- Ícones de funcionalidade;
- Ícones de menu;
- Ícone da aplicação (*launcher*);

Para a formulação dos ícones, utilizou-se como referência os *guidelines* fornecidos pela Google para aplicações Android. Os aspectos principais destes *guidelines* podem ser visualizados no segundo ponto do Anexo 4, onde para além de dimensões, efeitos e cores, também é possível ver alguns exemplos do que é considerado correcto e incorrecto para cada tipo de ícone. Relativamente aos três primeiros tipos de ícones (estado, funcionalidade, menu), foi definido que todos deveriam apresentar efeitos visuais semelhantes de forma a apresentarem alguma consistência entre eles. Para a formulação da maioria dos ícones recorreu-se a um *pack* de *templates,* também fornecidos pela Google que preservam o *design* e as texturas existentes nas aplicações nativas de Android, permitindo obter os diferentes estilos para cada categoria de ícone (por exemplo, *menu icon, list view icon, tab icon, launcher icon*). O processo de formulação dos ícones está descrito, passo a passo, no Anexo 5 deste documento.

Os **ícones de estado** são os ícones que representam o estado dos dispositivos (sensores e actuadores) presentes na rede do sistema B-Live Wireless. Para este tipo de ícones foram adicionadas 3 cores (vermelho, amarelo, verde) com o intuito de representar eficazmente o estado real dos dispositivos. Na Tabela 8 podem ser observados os ícones desenvolvidos para as duas aplicações, com uma pequena descrição, baseada no tipo de dispositivos a incluir no pré-piloto e no piloto, como foi referido no tópico 3.2.1 deste documento.

Tabela 8 – Ícones de estado

Ícone	Descrição	Ícone	Descrição
	Ar condicionado ligado		Ar condicionado desligad0
	*Buzzer* a emitir som		*Buzzer* sem emissão de som
	Interruptor pressionado		Interruptor à espera de ser pressionado
	Fumo detectado		Sem fumo detectado
	Dispositivo sob pressão		Dispositivo sem pressão
	Janela aberta		Janela fechada
	Porta aberta		Porta fechada
	Porta a fechar		Porta a abrir
	Lâmpada ligada		Lâmpada desligada

	Tomada ligada		Tomada desligada
	Persiana fechada		Persiana aberta
	Persiana a abrir		Persiana a fechar
	Movimento detectado		Sem movimento detectado
	Torneira aberta		Torneira fechada
	Medição de distância		Medição de temperatura
	Medição da luminosidade		Estado desconhecido ou dispositivo em inactividade

Os **ícones de funcionalidade** indicam o tipo de funcionalidade principal possível de utilizar na aplicação. Foram inicialmente definidas quatro funcionalidades básicas: Estado, Localização, Monitorização e Histórico. Contudo, apenas uma das funcionalidades foi implementada, a de Estado. Na Tabela 9 é possível observar os ícones de funcionalidade criados de acordo com as funcionalidades definidas previamente.

Tabela 9 – Ícones de funcionalidade

Ícones	Descrição
	Funcionalidade de Estado: permite visualizar e controlar o estado dos dispositivos presentes na rede
	Funcionalidade de Localização: permite visualizar a posição de uma pessoa ou objecto no interior da habitação
	Funcionalidade de Monitorização: permite monitorizar os sinais vitais de uma pessoa, tais como a frequência cardíaca, a temperatura ou a pressão arterial
	Funcionalidade de Histórico: permite visualizar os eventos ocorridos na rede num espaço temporal definido

Relativamente aos **ícones de menu**, estes servem para representar as opções contidas nos menus que surgem quando se pressiona o botão de "MENU" do dispositivo móvel. Nesta fase, nem todas a opções foram implementadas. Na Tabela 10 estão apresentados os ícones usados nos menus da aplicação.

Tabela 10 – Ícones de menu

Ícones	Descrição	Ícones	Descrição
	Opção para voltar ao início da aplicação		Opção para alterar o tema da aplicação (por exemplo, cores, estilos)
	Opção de escolha do tipo de ordenação na listagem dos dispositivos (alfabeticamente, última actualização)		Opção para escolha do tipo de visualização dos dispositivos (por exemplo, divisão, tipo de disp.)

	Opção para apresentação de ajuda		Opção para apresentação da informação sobre a aplicação
	Opção para actualização do estado dos disp. apresentados		Opção para efectuar configurações da aplicação
	Ícone para realizar operações relativas ao utilizador (por exemplo, ver informações, terminar sessão)		

Por último, os **ícones das aplicações**. Estes ícones representam a aplicação e são usados, por exemplo, no *home screen*[21] do sistema. O ícone pretende simbolizar uma habitação que utiliza tecnologia *wireless* no seu interior, que no fundo retrata o conceito do sistema B-Live Wireless.

Figura 27 – *Launcher icon* para a aplicação do pré-piloto

Figura 28 – *Launcher icon* para a aplicação do piloto

Imagens

Para além dos ícones, também foram desenvolvidas imagens para as aplicações do pré-piloto e do piloto para serem apresentadas nos vários *layouts* da aplicação. As imagens formuladas são os logótipos das aplicações, os logótipos das principais entidades envolvidas nos pilotos e as imagens dos botões de actuação dos dispositivos. Tal como para a produção dos ícones, as ferramentas base para o desenvolvimento das imagens foram o Adobe Illustrator[22] [63] e o Adobe Photoshop[23] [64].

[21] Análogo ao ambiente de trabalho em *desktops*.
[22] Versão *trial*: http://www.adobe.com/cfusion/tdrc/index.cfm?product=illustrator.
[23] Versão *trial*: http://www.adobe.com/cfusion/tdrc/index.cfm?product=photoshop.

Na Figura 29 e na Figura 30 são possíveis observar os logótipos para as duas aplicações. Sendo a aplicação denominada de "B-Live Home", considerou-se pertinente a inclusão do ícone da aplicação que simboliza uma habitação, para representar a palavra *Home*, em adição à palavra B-Live.

Figura 29 – Logótipo da aplicação para o pré-piloto (Micro I/O)

Figura 30 – Logótipo da aplicação para o piloto (ESSUA)

Na Figura 31, na Figura 32 e na Figura 33 estão representadas as principais entidades envolvidas na criação do pré-piloto e do piloto: o logótipo da Micro I/O que está incluído nas duas aplicações, o logótipo da ESSUA que apenas está inserido na aplicação do piloto, assim como a imagem com os ícones da Micro I/O e da Universidade de Aveiro. Esta imagem que contém apenas os ícones das entidades tem como intuito ser colocada nos *layouts* a utilizar pelos ecrãs de menor dimensão de modo a optimizar a área disponível dos *layouts*.

Figura 31 – Logótipo da Micro I/O

Figura 32 - Logótipo da ESSUA

Figura 33 – Imagem com os ícones da Micro I/O e da Universidade de Aveiro

Outro tipo de imagens formuladas foram as imagens para os botões de actuação dos dispositivos. Para cada aplicação optou-se por diferentes estilos de botão. Como se pode observar através da Tabela 11, foram desenvolvidas três imagens para os botões da aplicação do pré-piloto e duas para a aplicação do piloto. Para a aplicação do pré-piloto, no caso do dispositivo apenas permitir a comutação entre dois estados (por exemplo, aberto/fechado, ligado/desligado), são utilizadas as duas primeiras imagens: verde para activar/abrir/ligar e vermelho para desactivar/fechar/desligar. Se o dispositivo tiver associado vários comandos de actuação, como no caso do *buzzer* em que podem ser escolhidos vários sons a emitir, ou no caso do ar condicionado em que se pode escolher uma temperatura específica, é utilizada a

imagem azul. Contudo, esta abordagem não é a mais apropriada pois não fornece *feedback* visual quando é pressionado, tal como é recomendado nas boas prática já referidas. Por isso, na aplicação do piloto decidiu-se simplificar estes botões, apresentando o mesmo para actuar sobre todos os dispositivos, e adicionalmente, existir uma imagem para o estado em que o botão está a ser pressionado (duas últimas imagens da tabela).

Tabela 11 – Botões de actuação

Imagem	Descrição
	Imagem do botão de actuação para activar o dispositivo (pré-piloto)
	Imagem do botão de actuação para desactivar o dispositivo (pré-piloto)
	Imagem do botão de actuação para dispositivos especiais (pré-piloto)
	Imagem do botão de actuação no estado normal (piloto)
	Imagem do botão de actuação no estado pressionado (piloto)

Layouts

Em Android os *layouts* são a arquitectura do UI num dado instante. Estes podem ser definidos de duas formas: recorrendo a vocabulário XML ou instanciando os elementos que compõem o *layout* em *runtime*. A primeira opção permite uma separação mais evidente da apresentação da aplicação relativamente ao código que controla o seu comportamento. No segundo caso, a definição da apresentação e do comportamento dos seus elementos é toda realizada em código Java, podendo tornar o código difícil de perceber. Neste sentido, no desenvolvimento das duas aplicações optou-se por usar a primeira opção para a definição dos vários *layouts*.

De seguida, são apresentados os *layouts* das aplicações, baseado na definição da aplicação apresentada na secção 4.2. Adicionalmente, são apresentados alguns *layouts* para as situações em que o dispositivo se encontra orientado horizontalmente (*landscape*). Estão

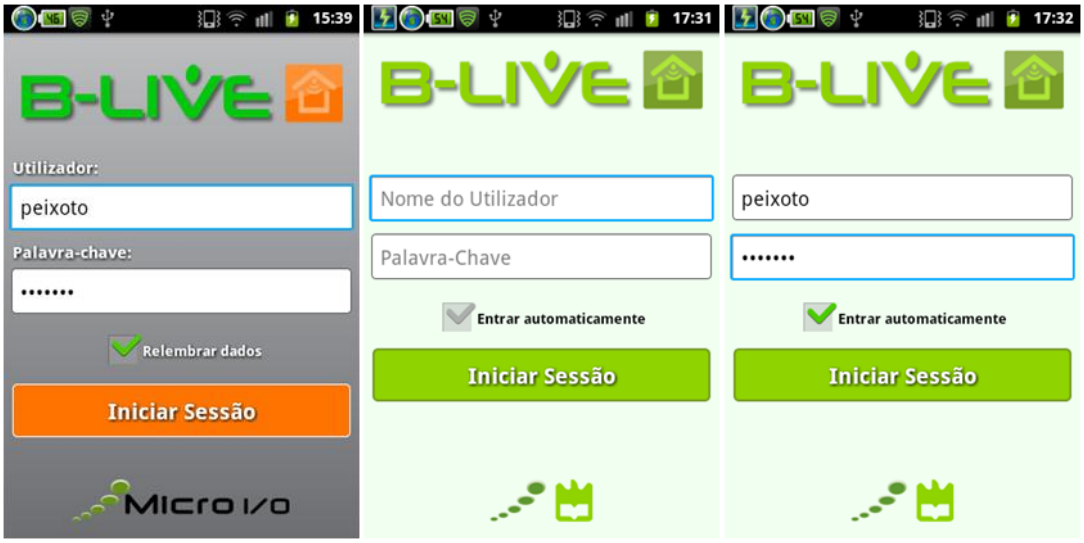
apresentados em conjunto os *layouts* para as duas aplicações, evidenciando-se as diferenças entre elas.

Na Figura 34 é possível observar os *layouts* para as duas aplicações aquando da sua inicialização. Na Figura 34-A, é possível observar a mensagem de erro apresentada na aplicação do pré-piloto aquando da verificação da ligação ao servidor. Como foi referido, no pré-piloto a comunicação com o servidor é realizada através de um protocolo proprietário sobre um *socket* TCP, pelo que quando a aplicação está a inicializar, começa por verificar se consegue criar o canal de comunicação com o servidor através de *socket*. No caso de não ser possível, apresenta a mensagem da imagem e clicando em "OK", a aplicação é terminada. Caso contrário, depois de inicializar a aplicação, é apresentado o *layout* de autenticação no sistema.

A **B** **C**

Figura 34 – Aplicações: *Layouts* de inicialização

Para a autenticação no sistema, a estrutura dos *layouts* das duas aplicações são semelhantes como se pode verificar na Figura 35. A principal diferença está relacionada com a *checkbox* apresentada. Para o pré-piloto, ao seleccionar-se a *checkbox* implica que da próxima vez que se iniciar a aplicação, os campos "Utilizador" e "Palavra-chave" já estarão preenchidos, enquanto que para o piloto, se a *checkbox* estiver seleccionada, quando a aplicação for iniciada, aplicação fará a autenticação automaticamente sem apresentar o *layout* de autenticação. Esta alteração entre as duas aplicações deve-se ao facto de se ter constatado que para a generalidade das aplicações móveis o último é o procedimento mais utilizado (por exemplo, a aplicação do Facebook). Contudo, esta funcionalidade para a aplicação do piloto ainda não está implementada.



A B C

Figura 35 – Aplicações: *Layouts* de autenticação

Após se ter efectuado a autenticação no sistema com sucesso, é apresentado o *layout* com os botões associados às diversas funcionalidades da aplicação (Figura 36). Os botões que se encontram em tons de cinzento significam que essas funcionalidades ainda não estão implementadas. Neste *layout* também é apresentado a identificação do utilizador autenticado. Tal como foi referido no capítulo sobre as boas práticas para o desenvolvimento de aplicações, a interacção entre o utilizador e o UI deve fornecer sempre um feedback visual ao utilizador. Este comportamento pode ser observado, por exemplo na Figura 36-C. Nesta situação, o utilizador pressionou na identificação, mudando a cor envolvente durante o período em que o utilizador pressiona aquela zona do ecrã. Esta técnica é também utilizada para outros controlos, como por exemplo, os botões.

A B C

Figura 36 – Aplicações: *Layouts* das funcionalidades

Ao seleccionar-se a funcionalidade de "Estado", é apresentada a lista dos dispositivos existentes na rede, como se pode observar na Figura 37. No caso da aplicação do piloto, que opera com *Web Services*, é apresentada uma mensagem a informar que está a obter a informação. Na Figura 37-B e na Figura 37-C é possível verificar que os dispositivos são apresentados sob a forma de lista, em que cada item da lista corresponde a um dispositivo. O item é constituído por:

- Um ícone, que representa o estado do dispositivo;
- O nome do dispositivo;
- Uma mensagem relativamente ao estado no dispositivo;
- Um botão de actuação no caso de o dispositivo ser um actuador.

A principal diferença entre as listas das duas aplicações encontra-se no aspecto do botão de actuação. Para a aplicação do piloto optou-se por uniformizar e simplificar os botões.

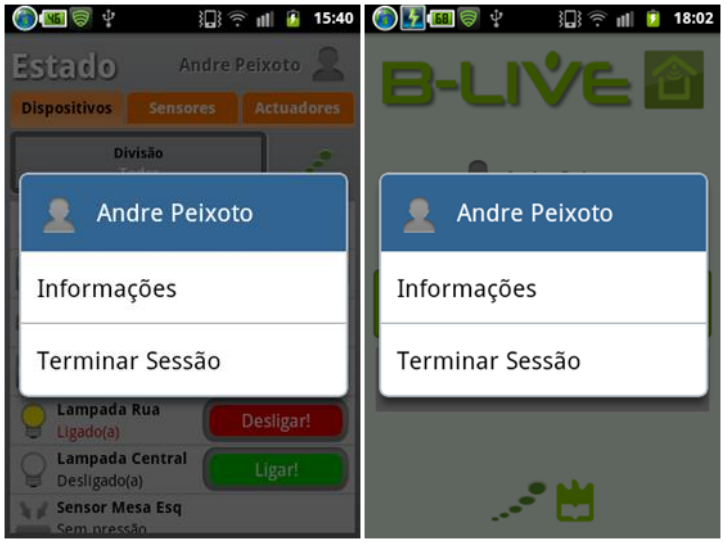
Para além da lista de dispositivos, é apresentada a identificação do utilizador, um conjunto de *tabs* que permite filtrar os sensores e os actuadores. Por baixo das *tabs*, encontram-se os controlos que permitem refinar o tipo de filtragem (por exemplo, por sector divisão).

A **B** **C**

Figura 37 – Aplicações: *Layouts* da funcionalidade de Estado

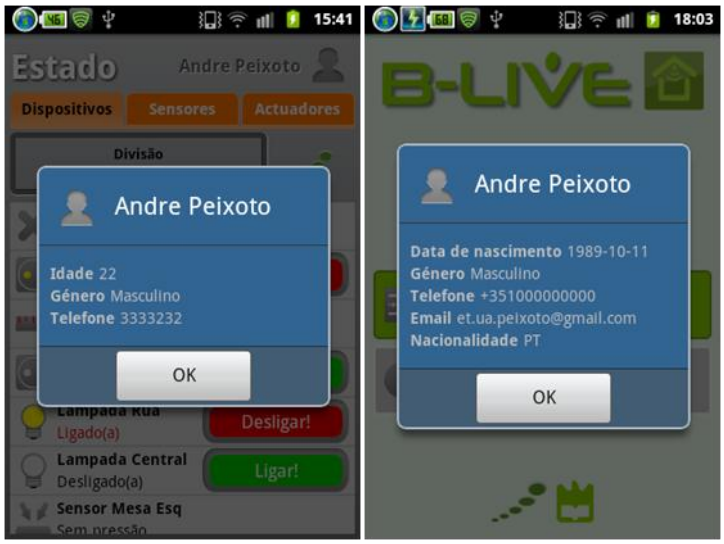
Como foi referido, tanto o *layout* com as funcionalidades da aplicação como o do estado dos dispositivos apresenta a identificação do utilizador. Ao clicar-se nesta identificação são apresentadas duas opções relativas ao utilizador: a possibilidade de visualizar as informações associadas ao utilizador na base de dados do sistema, ou de terminar a sessão. A Figura 38 representa estes *layouts*. Ao seleccionar-se a opção de visualização das informações, é apresentada uma mensagem com a respectiva informação, como se pode observar na Figura 39. Ao escolher-se a opção para terminar sessão, é desencadeado o processo de *logout* do sistema, voltando a surgir o *layout* de autenticação

apresentado na Figura 35, não sendo possível a partir deste momento obter mais informação do sistema até se iniciar uma nova sessão.



A **B**

Figura 38 – Aplicações: *Layouts* com opções relativas ao utilizador



A **B**

Figura 39 – Aplicações: *Layouts* com informações do utilizador

A B

Figura 40 – Aplicações: *Layouts* com resultado da filtragem para sensores

Através da Figura 40 e da Figura 41 é possível observar a filtragem feita à lista dos dispositivos recorrendo às *tabs* do *layout*, permitindo que sejam apresentados os sensores e actuadores de forma separada.

A B

Figura 41 – Aplicações: *Layouts* com resultado da filtragem para actuadores

Como pode ser observado na Figura 42-A, é possível visualizar um menu que é accionado quando é pressionado o botão de "MENU" do dispositivo móvel. Este menu permite, por exemplo, efectuar algumas acções sobre a lista dos dispositivos. A Figura 42-B apresenta as opções de filtragem quando se escolhe a opção de "Visualização" no menu. No caso da aplicação para o piloto, esta opção pode ser acedida através do controlo que se encontra abaixo das *tabs*, do lado esquerdo. Esta alteração entre as duas aplicações visa simplificar o acesso a este tipo de funcionalidade (filtragem), pois desta forma não é necessário invocar o menu.

A B C

Figura 42 – Aplicações: *Layouts* com definição do tipo de filtro 1

A maioria dos actuadores existentes na rede apenas permite a comutação entre dois estados, como por exemplo, as lâmpadas que apenas podem ser ligadas ou desligadas. Por isso, quando é realizado um pedido de actuação através da pressão de um botão (de actuação), o comando associado é enviado imediatamente para servidor. Contudo, no caso do pré-piloto, existem dispositivos considerados especiais que podem comutar entre vários estados. São os casos do *buzzer* e do ar condicionado. Na Figura 43, pode-se observar a lista de opções que é possível escolher para cada um dos dispositivos, quando é pressionado o respectivo botão de actuação. Só depois de seleccionar uma das hipóteses é que é enviado o pedido para o servidor.

A B

Figura 43 – Aplicações: *Layouts* para actuação sobre dispositivos multi-estado

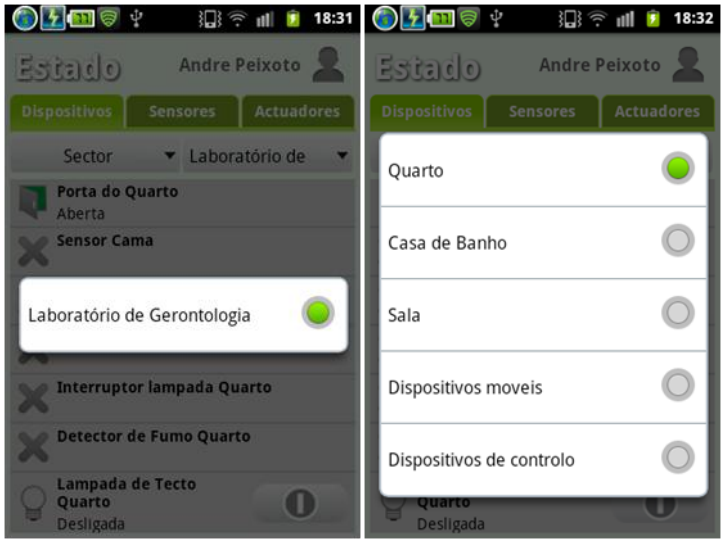
Na aplicação do pré-piloto, é também possível refinar a filtragem, como está apresentado na Figura 44. Ao clicar no controlo que se encontra abaixo das *tabs* do *layout*, é

apresentada a lista de opções existentes para a filtragem, no exemplo das imagens, por tipo de dispositivo. A Figura 44-C demonstra o resultado da filtragem quando se selecciona a opção "Lâmpada".

A **B** **C**

Figura 44 – Aplicações: *Layouts* com definição do tipo de filtro 2

Para a aplicação do piloto também existe suporte para esta funcionalidade, como é possível verificar na Figura 45.



A **B**

Figura 45 – Aplicações: *Layouts* com definição do tipo de filtro 3

Recuperando o menu apresentado na Figura 42-A, outra das acções possíveis é o da ordenação da lista dos dispositivos. Na Figura 46-A é possível verificar os tipos de ordenações possíveis (por ordem alfabética ou por última actualização), e na imagem Figura 46-B verifica-se o resultado obtido quando é escolhida a opção de ordenação alfabética.

A **B**

Figura 46 – Aplicações: *Layouts* com definição do tipo de ordenação

Cada item da lista é passível de ser pressionado, sendo apresentada uma mensagem com informação adicional relativa ao dispositivo escolhido. Informações como a sua localização, última actualização, e até o nível da bateria do componente responsável pela transmissão da informação para a rede, são possíveis de visualizar, tal como se pode observar na imagem Figura 47-A. Para além disto, ao clicar-se sobre o logótipo da Micro I/O que se encontra no *layout*, é possível ter acesso a algumas informações relativas à aplicação (Figura 47-B).

A **B**

Figura 47 – Aplicações: *Layouts* com informação sobre os dispositivos e sobre a aplicação

Para finalizar, sendo um dos requisitos o suporte para variações da orientação de ecrã, todos os *layouts* apresentados acima tem a versão correspondente para *landscape*. A disposição dos diferentes elementos que compõem o UI tem de ser necessariamente

diferente, pois para esta situação o ecrã apresenta maior largura e menor altura. Por isso, foi necessário redefinir a disposição destes elementos com o intuito de adaptar o UI a estas características. Na Figura 48 é possível observar os *layouts* principais tendo em vista a utilização para orientação horizontal.

Figura 48 – Aplicações: *Layouts* em *landscape*

Como foi referido, as aplicações devem adaptar-se a vários tipos de ecrã. Todos os *layouts* apresentados anteriormente foram obtidos de testes realizados no *smartphone* Samsung Galaxy Gio em que o tamanho do ecrã é considerado "*large*" e a densidade "*medium*". No Anexo 6 é possível verificar que as aplicações suportam outros tipos de ecrã.

4.3.2 *Core*

Como referido, a linguagem geralmente utilizada para o desenvolvimento de aplicações em Android é o Java. Esta é a motivação de se ter usado esta linguagem para o

desenvolvimento do código fonte das aplicações, onde se define a estrutura, o comportamento dos controlos do UI, a interacção entre os *layouts* com os utilizadores, a persistência de dados e o processo de comunicação com o servidor do sistema B-Live Wireless.

Para a definição destas componentes das aplicações, recorreu-se à linguagem de modelação normalizada *Unified Modeling Language* (UML) que é especialmente útil para o desenvolvimento de *software* orientado a objectos. O UML é uma linguagem gráfica que permite a visualização, especificação, criação e documentação dos artefactos de sistemas de *software*. A versão UML 2.x é composta por treze tipos de diagramas que podem ser classificados da seguinte forma [65]:

- *Structure diagrams*: tipos de diagramas que retratam elementos especificados do sistema que são independentes do instante temporal (*class, composite structures, component, deployement, object* e *package diagrams*);
- *Behavior diagrams*: diagramas que definem o comportamento do sistema (*activity, state* machine e *use case diagrams*);
- *Interaction diagrams*: são um subconjunto dos *behavior diagrams* e retratam a interacção entre objectos (*communication, interaction overview, sequence* e *timing diagrams*);

De seguida, é especificado o comportamento e a estrutura das aplicações, assim como a correspondente interacção com servidor. Neste sentido, recorre-se a alguns dos diagramas mencionados para os representar graficamente. Os diagramas utilizados são: *activity diagram*, *class diagram* e *sequence diagram*. Os diagramas são complementados com uma descrição. As definições mais pormenorizadas dos conceitos de desenvolvimento em Android utilizados na descrição são remetidas para os anexos do documento. A especificação das aplicações do pré-piloto (Micro I/O) e do piloto (ESSUA), é efectuada de forma separada, devido às diferenças na sua implementação.

Pré-piloto (Micro I/O)

De forma a representar o comportamento da aplicação, foi elaborado um *activity diagram*, que pode ser observado na Figura 49. O diagrama tem como objectivo representar graficamente aspectos como a navegação da aplicação entre os vários *layouts* que constituem o seu UI e as acções resultantes da interacção com o utilizador.

Ao iniciar a aplicação, é apresentado o *layout* de inicialização através de uma *activity* (consultar a definição no ponto 4 do Anexo 4). Este *layout* é apresentado durante alguns segundos para que possam ser efectuadas algumas operações de inicialização. Como está descrito no ponto 5 do Anexo 4, não devem ser efectuadas operações que bloqueiem o UI da aplicação. Por isso, é iniciada uma nova *thread* que é executada em *background*. A *thread* principal é responsável por actualizar a barra de progresso que indica a inicialização da aplicação. Durante a sua execução, a *thread* em *background* analisa ficheiros XML[24] disponíveis nos *resources* da aplicação, para obter (e inicializar) os seguintes dados: endereço IP do servidor, porto do *socket* TCP e índices que permitem o acesso correcto à base de dados do servidor (este aspecto é abordado com maior detalhe mais à frente). Esta *thread* tenta abrir o

[24]Faz o *parsing* dos ficheiros.

socket TCP com o servidor e no caso de não ser possível, é apresentada uma mensagem no UI a informar que a ligação com o servidor falhou. O utilizador terá então que terminar a aplicação e voltar posteriormente, após ter verificado se está na mesma rede em que se encontra o servidor. Se pelo contrário, a ligação for estabelecida com sucesso, é lançada uma nova *thread* para manipular as comunicações com o servidor (este aspecto não se encontra representado no diagrama, mas é abordado com maior pormenor à frente).

Após a inicialização da aplicação, é iniciada uma nova *activity* com recurso a *intents* (consultar definição no ponto 6 do Anexo 4). Esta *activity* tem o propósito de manipular o processo de autenticação no sistema B-Live Wireless. Recorrendo ao mecanismo de *shared preferences* para persistência de dados (ponto 7 do anexo 4) é verificado se existem dados guardados do nome do utilizador e palavra-chave em utilizações anteriores da aplicação. Se existirem, esses dados são apresentados nos respectivos campos. Caso contrário, o utilizador preenche os campos com os seus dados. Ao clicar no botão para iniciar sessão, é verificado se os campos estão preenchidos e em caso afirmativo, é iniciado o processo de comunicação com o servidor no sentido de autenticar a aplicação no sistema, recorrendo a uma *thread* criada para o efeito. Se o acesso for negado, é apresentada uma mensagem de erro e o *layout* continua a ser o mesmo. Se o acesso for permitido, guarda os dados no caso de a *checkbox* associada à funcionalidade "Relembrar dados" estiver seleccionada, iniciando de seguida uma nova *activity*.

O *layout* apresentado após a autenticação possibilita ao utilizador escolher uma das principais funcionalidades da aplicação[25]. Contudo apenas uma se encontra implementada até ao momento: monitorização e controlo do estado dos dispositivos. Quando o utilizador escolhe esta funcionalidade, é apresentado o UI correspondente. Adicionalmente, o utilizador pode aceder a funcionalidades mais específicas relacionadas com a sua conta B-Live. Se o utilizador clicar sobre a identificação, é apresentada uma mensagem com duas funcionalidades possíveis: visualizar a informação associada ao seu perfil (dados estão guardados na base de dados do servidor e são obtidos durante o processo de autenticação) e terminar sessão. Se for escolhida a primeira hipótese, é apresentada uma nova mensagem com os respectivos dados do utilizador. Se a opção escolhida for a de terminar sessão, é realizado o processo de *logout*, voltando a aplicação para o *layout* de autenticação.

[25] As funcionalidades previstas são: Estado, Localização, Monitorização e Histórico.

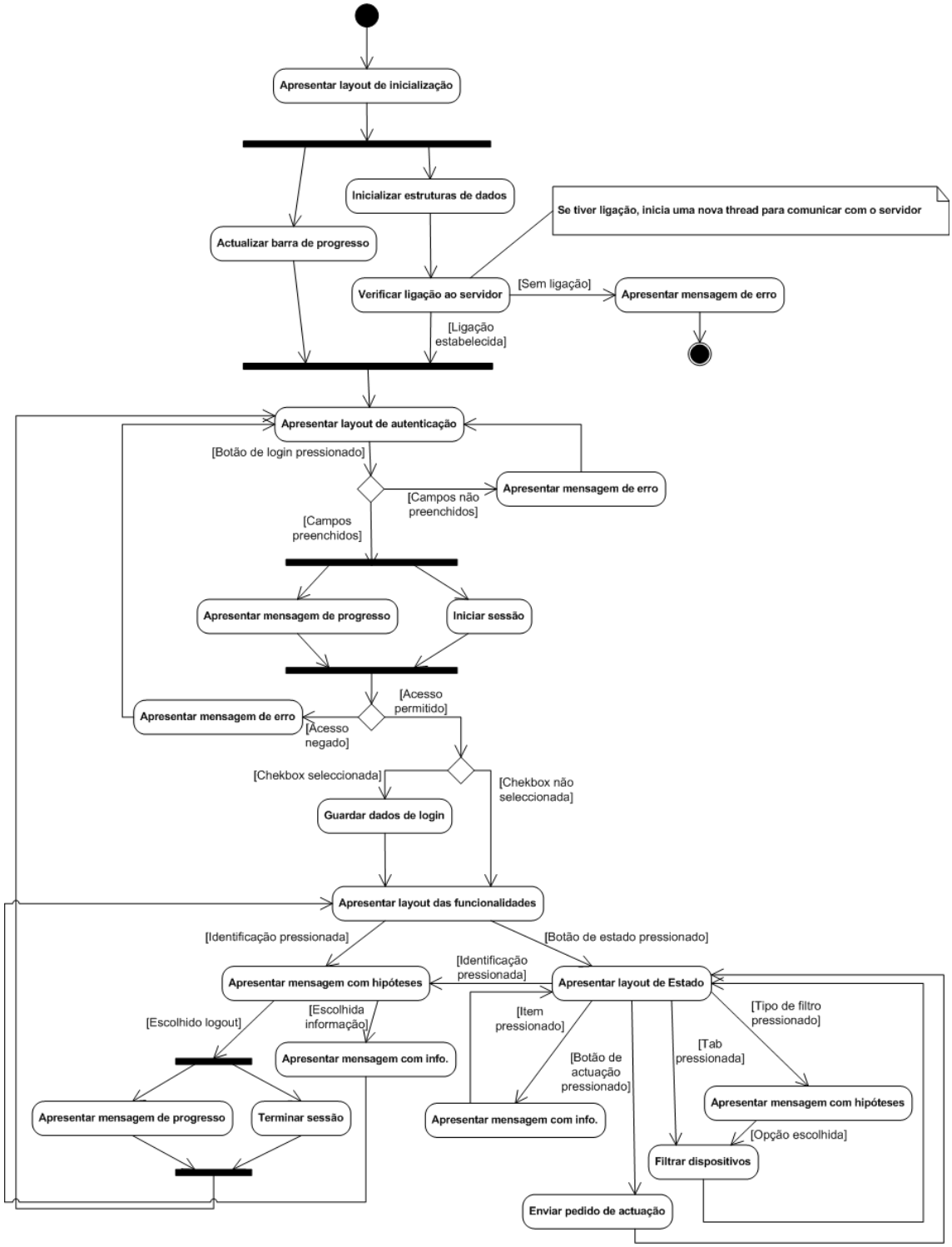


Figura 49 – *Activity Diagram* para a aplicação do pré-piloto

Relativamente à *activity* associada ao layout da funcionalidade de "Estado", a sua principal característica é a capacidade de apresentar a lista dos dispositivos existentes no sistema, com a possibilidade de se efectuar determinadas acções sobre a lista, de forma a alterar o conjunto de dispositivos listados. A principal acção passível de ser executada é a filtragem da lista. O *layout* é constituído por 3 *tabs* (Dispositivos, Sensores, Actuadores) que permitem filtrar a lista de acordo com este tipo de classificação. Outro tipo de filtragem está

relacionado com a localização ou o tipo de dispositivo (por exemplo, tomada). Ao clicar-se sobre um dos dispositivos, é apresentada uma mensagem com informação relativa a esse dispositivo, como por exemplo, localização, última alteração de estado e nível de bateria. Quando um botão de actuação é pressionado, é enviada uma mensagem para o servidor, com o respectivo pedido de actuação. Adicionalmente, existe a possibilidade de ordenar a lista dos dispositivos por ordem alfabética ou pela data de última alteração de estado. Contudo, esta funcionalidade está apenas acessível a partir do menu utilizado para esta *activity*, pelo que não se encontra representada no diagrama.

O diagrama da Figura 49 não faz alusão explícita ao ciclo de vida da aplicação. No ponto 4 do Anexo 4 este aspecto é abordado com mais detalhe.

Como foi referido, o *activity diagram* é utilizado para modelar o comportamento dinâmico da aplicação. De seguida, é apresentado o *class diagram* (Figura 50), para representar a modelação estática da aplicação. Os atributos e métodos de algumas classes foram omitidos, com o intuito de simplificar o diagrama.

A aplicação é composta por quatro *activities*, denominadas por "*AndroidUIActivity*"[26], "*LoginActivity*"[27], "*FunctionalitiesActivity*"[28] e "*StateTabActivity*"[29]. Estas classes são generalizações (herdam atributos e métodos) da superclasse "*Activity*"/"*TabActivity*" que são componentes da aplicação (ponto 4 do Anexo 4). Durante o ciclo de vida da aplicação, as diferentes *activities* podem necessitar de aceder a dados em comum, por exemplo, a "*FunctionalitiesActivity*" e a "*StateTabActivity*" necessitam da informação do utilizador. Uma das formas para passar dados de uma *activity* para outra é recorrendo ao contexto da aplicação, que pode ser utilizado como um *singleton*[30]. A classe "*DataApp*" é uma subclasse da classe *Application* e é esta classe ("*DataApp*") que é usada como *singleton*, permitindo a partilha de dados entre as *activities*. Como se pode observar no diagrama, as *activities* têm associadas a classe "*DataApp*", para poder aceder aos dados partilhados. Esta classe contém como atributo um objecto do tipo "*BliveAPI*", sendo este o objecto que se pretende partilhar.

A classe "*BliveAPI*" contém referências de um conjunto de objectos dos quais se destacam:

- *"(BliveUser) user*": Este objecto contém a informação do utilizador que se encontra autenticado no sistema (por exemplo, nome, idade, telefone, nome de utilizador, palavra-chave);
- "*(BliveSystem) info*": Este objecto é utilizado para guardar a informação, de forma hierárquica, relativa ao sistema B-Live. A constituição desta classe é descrita com maior detalhe mais à frente;
- "*(BliveClient) connection*": Este objecto manipula a comunicação com o servidor. Contém a informação relativa ao *socket* de comunicação e a classe implementa a interface *Runnable*, pelo que contém um método (*run()*) que pode ser executado no contexto de uma *thread* em *background*.

[26] Responsável por apresentar o *layout* de inicialização.
[27] Responsável por apresentar o *layout* de autenticação.
[28] Responsável por apresentar o *layout* das funcionalidades.
[29] Responsável por apresentar o *layout* da funcionalidade de Estado.
[30] Padrão de *design* de *software* em que a instanciação de uma classe é limitada a um objecto.

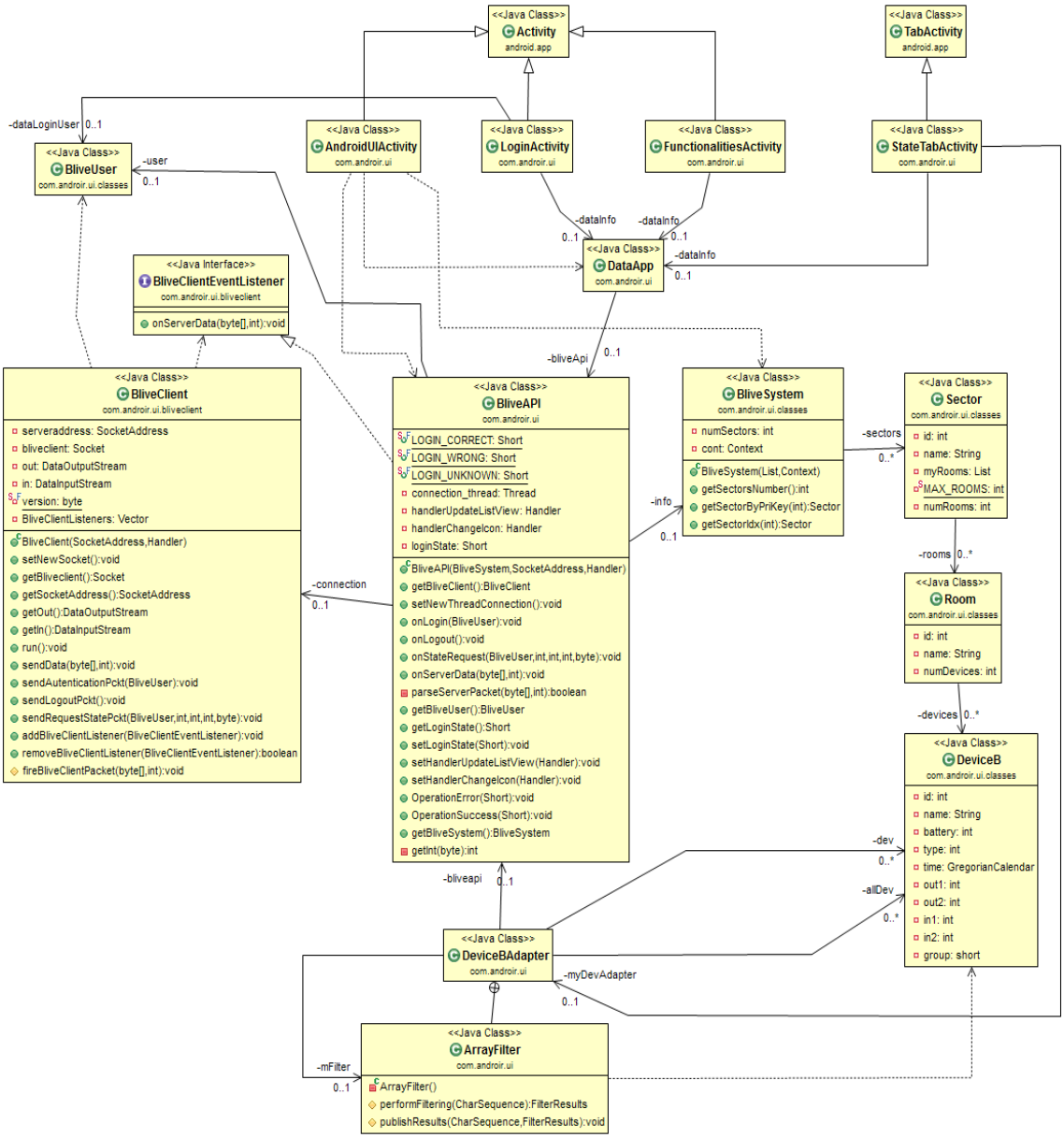


Figura 50 – *Class Diagram* para a aplicação do pré-piloto

A interface "*BliveClientEventListener*" permite que, quando detectados eventos de chegada de mensagens vindas do servidor, seja desencadeado o processo de *parsing* das tramas (formato proprietário), actualizando as estruturas de dados adequadas.

Os dispositivos presentes no sistema B-Live Wireless são identificados por 4 atributos: sector, divisão, grupo e número identificativo. Por exemplo, num sistema real implementado numa habitação, um sector seria o 1º andar, uma divisão a cozinha, o grupo lâmpada e o número identificativo poderia ser o número 3. Através destes atributos é possível localizar e identificar o dispositivo no sistema. De acordo com esta hierarquia, a classe "*BliveSystem*" contém um *array* de sectores (classe "*Sector*"), em que cada sector contém um *array* de divisões (classe "*Room*") e cada divisão contém um *array* de dispositivos (classe "*DeviceB*"). Relativamente aos dispositivos, o estado do mesmo é reflectido nos atributos "*in1*", "*in2*", "*out1*" ou "*out2*", dependendo do tipo de dispositivo (sensor ou actuador).

Para a manipulação da lista de dispositivos a apresentar, a "*StateTabActivity*" utiliza a classe "*DeviceBAdapter*" (é uma subclasse da classe *ArrayAdapter*) que é responsável por interligar os dados que se pretende apresentar, neste caso, provenientes dos objectos do tipo "*DeviceB*", com os elementos do UI onde esses dados são apresentados, ou seja, no respectivo item da lista. Por exemplo, quando existe uma alteração de estado, o ícone do dispositivo pode ter de ser alterado. Nesta situação, a classe "*DeviceBAdapter*" é constituída por métodos que permitem actualizar o ícone imediatamente após a informação sobre o dispositivo ter sido recebida pela aplicação. Esta classe contém uma referência para um *ArrayList* com todos os dispositivos presentes no sistema ("*allDev*") e outra referência para um *ArrayList* que contém apenas os dispositivos que estão apresentados num determinado momento ("*dev*"). O tamanho de "*dev*" que pode ser menor ou igual ao número total de dispositivos. Isto deve-se ao facto de poderem ter sido efectuadas operações de filtragem, por exemplo, para serem apresentados apenas dispositivos do tipo sensor, e operações de ordenação. Para realizar operações de filtragem, a classe "*DeviceBAdapter*" recorre à classe interna "*ArrayFilter*". As operações de filtragem são realizadas de forma assíncrona e são executadas numa *thread* em *background* criada pelo sistema operativo, impedindo deste modo que o UI bloqueie.

Como foi referido, a informação relativa ao sistema é obtida através de um *socket* TCP, em que as tramas trocadas entre a aplicação cliente (AC) e a aplicação servidora (AS) seguem um protocolo proprietário da Micro I/O. De seguida, é descrito o tipo de mensagens e a sequência de mensagens trocadas ao longo do ciclo de vida a aplicação, recorrendo ao *sequence diagram* apresentado na Figura 51.

A troca de mensagens inicial entre a aplicação e o servidor está relacionado com o processo de autenticação da aplicação cliente no sistema B-live. Este processo consiste no envio de uma mensagem síncrona[31] de autenticação (quando é pressionado o botão de *login*) por parte da AC contendo o nome de utilizador e a palavra-chave inseridos pelo utilizador. A resposta por parte da AS pode ser de dois tipos:

- No caso de o acesso não ser autorizado, é recebida uma mensagem de erro, e o utilizador terá de tentar autenticar-se novamente.
- Caso contrário, é retornada a informação correspondente ao utilizador, sendo actualizado o objecto do tipo *BliveUser*.

Após a autenticação ter sido efectuada com sucesso, a AS envia uma mensagem (assíncrona) de sincronização com a informação relativa à identificação (nome) e ao estado de todo o sistema naquele instante. Esta informação é utilizada para actualizar as estruturas de dados anteriormente inicializadas (*BliveSystem, Sector, Room* e *Device*).

Depois da sincronização, a AC está apta a receber mensagens de ocorrência de eventos na rede de sensores e actuadores. Sempre que ocorre um evento na rede, essa informação é processada e armazenada no servidor. Através deste tipo de mensagens (assíncrona), a AS informa a AC da ocorrência do evento, com a AC a actualizar o respectivo objecto do tipo *Device* para o estado actual.

Como referido, o utilizador pode efectuar pedidos de actuação direccionados aos dispositivos actuadores. Quando o utilizador pressiona o botão de actuação de um dispositivo, é enviada uma mensagem (síncrona) de alteração de estado. No caso de o

[31] Bloqueante, ou seja, a AC só continua após a AS processar a mensagem.

servidor ter enviado com sucesso o pedido para a rede de sensores e actuadores, é retornada uma mensagem a indicar que o pedido foi enviado para a rede. Caso contrário, é enviada uma mensagem de erro.

O utilizador tem também a possibilidade de terminar sessão. Este processo consiste no envio de uma mensagem (síncrona) de *logout* para o servidor. A resposta indica que a operação foi concluída com sucesso. A partir deste instante, a AC deixa de receber informação relativa ao sistema, a não ser que o utilizador volte a autenticar-se com sucesso.

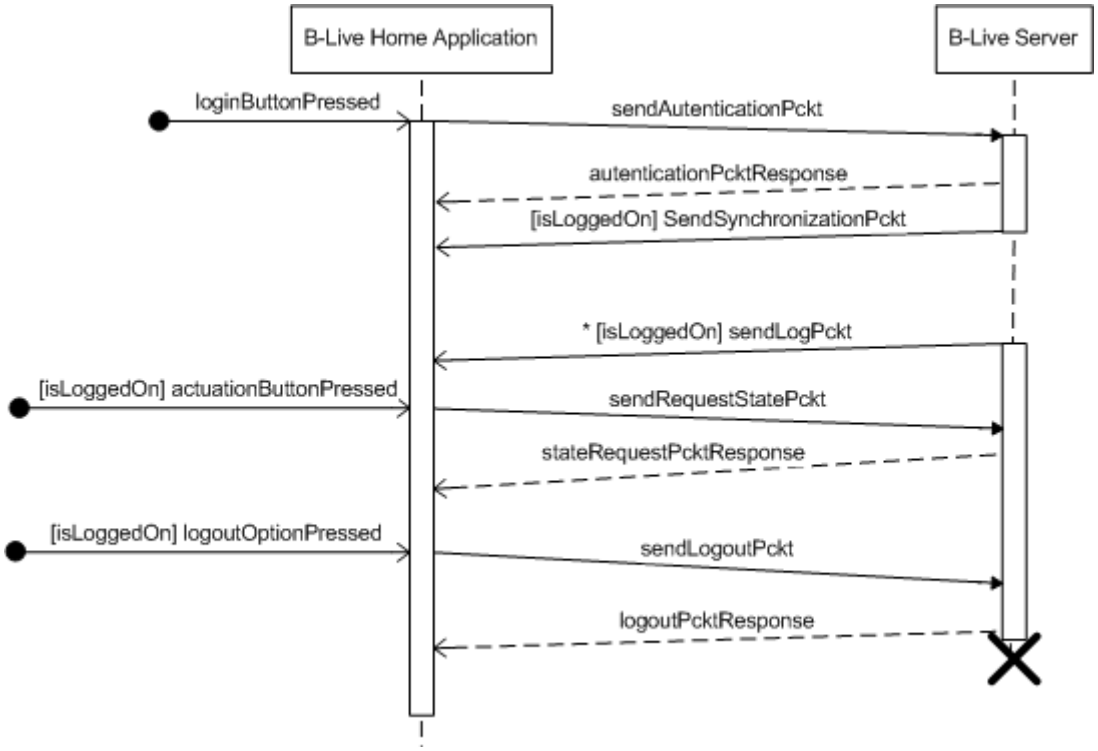


Figura 51 – *Sequence Diagram* da comunicação entre o servidor e a aplicação do pré-piloto

Piloto (ESSUA)

Como mencionado anteriormente, para a aplicação do piloto, a comunicação com o servidor do sistema B-Live é baseada em *Web Services*. Este facto exigiu uma reformulação do código fonte da aplicação do pré-piloto para que fosse possível comunicar com o servidor baseado nesta tecnologia. Ao nível da navegabilidade, isto é, da transição entre *layouts* devido à interacção com utilizador, as alterações não foram muito significativas. Isto significa que os *activity diagrams* das duas aplicações são semelhantes. As diferenças mais perceptíveis encontram-se nas estruturas de dados (*class diagram*) e na comunicação com o servidor (*sequence diagram*), como se verá adiante.

Na Figura 52 é possível observar o *activity diagram* para a aplicação do piloto. Numa fase inicial do desenvolvimento optou-se por não começar por apresentar um *layout* de inicialização da aplicação, uma vez que não é necessário inicializar previamente as estruturas

de dados que irão conter a informação de sistema B-Live. Esta informação é obtida através da invocação dos *Web Services* (WS) numa fase posterior do ciclo de vida da aplicação, como será explicado adiante. Devido ao facto do consumo dos diversos WS disponibilizados resultar de um processo em que a aplicação cliente envia uma mensagem (no formato adequado) e recebe outra mensagem como resposta, a verificação da conectividade com o servidor será realizada no momento de cada invocação de um WS. Em caso de falha no consumo do WS, é apresentada uma mensagem de erro.

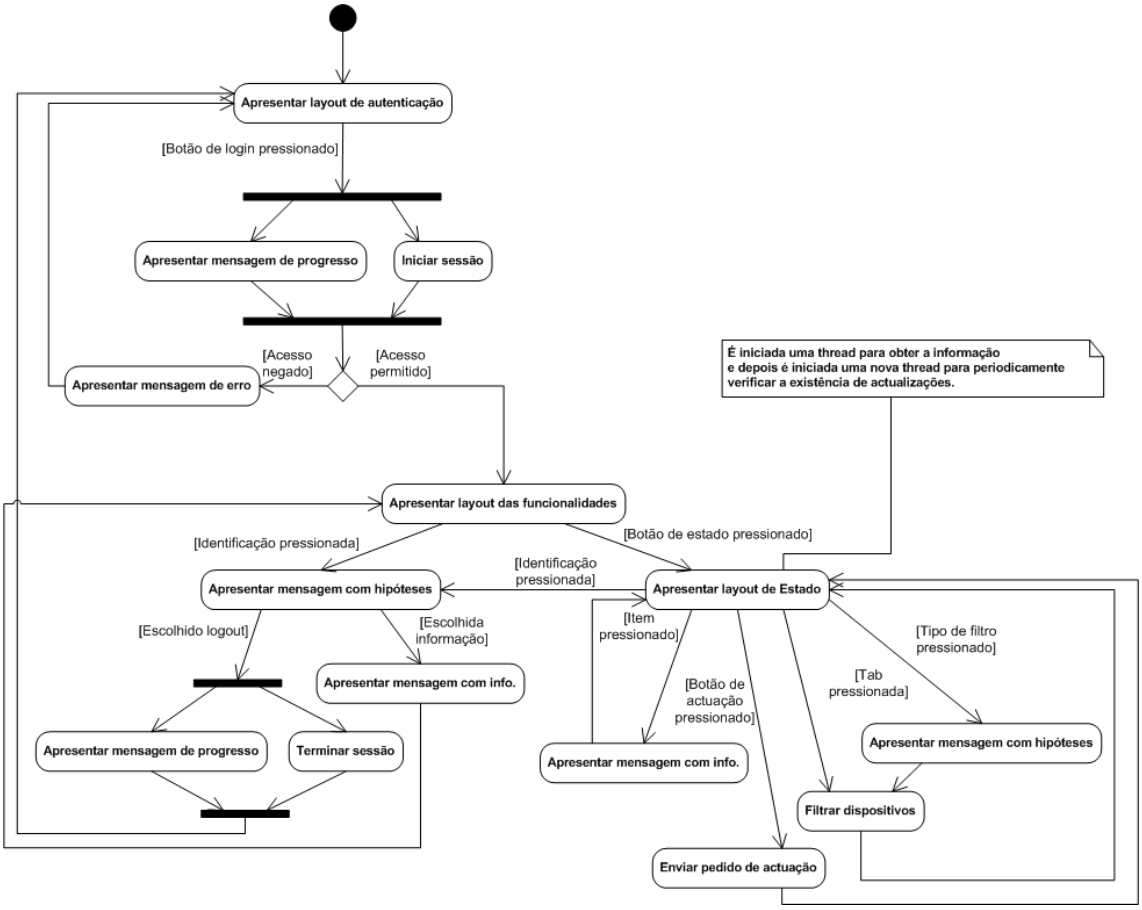


Figura 52 - *Activity Diagram* para a aplicação do piloto

Neste sentido, o primeiro *layout* a ser apresentado é o de autenticação. O UI apresentado durante o processo de autenticação é semelhante ao da aplicação do pré piloto. A diferença está na comunicação com o servidor. Relativamente à funcionalidade de "Entrar automaticamente", esta não se encontra ainda implementada. Ao receber a resposta, a aplicação verifica se o acesso é negado ou permitido. No caso de ser negado, é apresentada uma mensagem de erro ao utilizador, tendo o mesmo de voltar a tentar a autenticação. Na situação em que o acesso é permitido, o *layout* das funcionalidades da é apresentado.

Este *layout* apresenta as mesmas possibilidades de interacção com o utilizador que o implementado na aplicação do pré-piloto. O utilizador pode clicar no campo identificação e escolher a opção para verificar a sua informação pessoal, ou seleccionar a opção de terminar a sessão, voltando assim para o *layout* de autenticação. O utilizador pode optar aceder à funcionalidade de "Estado", sendo neste caso iniciada uma nova *activity* responsável por apresentar e manipular o *layout* da funcionalidade de "Estado".

Durante a criação da *activity*, é iniciado o processo de obtenção da informação do sistema. Para isso, é criada uma *thread* que é executada em *background*, responsável por invocar os WS apropriados e inicializar as estruturas de dados adequadas para armazenar a informação. Durante este período de tempo em que a informação está a ser recolhida, é apresentada uma mensagem a indicar que a aplicação se encontra a obter os dados. Ao finalizar o processo de obtenção da informação, os dispositivos são listados no UI e é imediatamente iniciada uma nova *thread* responsável por consumir periodicamente um serviço disponibilizado que permite obter a indicação de quais os dispositivos que foram actualizados num determinado intervalo de tempo. Mais à frente serão apresentados pormenores adicionais sobre este processo. A partir do momento em que os dispositivos são apresentados, a interacção entre o utilizador e o UI é em tudo semelhante ao que foi descrito para a aplicação do pré-piloto, com o utilizador a poder utilizar as mesmas funcionalidades (filtragem, ordenação e actuação).

De seguida é apresentado o diagrama de classes (Figura 53) da aplicação relativa ao piloto do sistema B-Live Wireless. Neste diagrama estão representadas as principais classes que constituem o modelo estático da aplicação. Também para este diagrama optou-se por não apresentar os atributos e métodos associados a algumas classes no sentido de simplificar o diagrama. Em adição às classes apresentadas, foram implementadas outras com o objectivo específico de manipular a interacção com o servidor através dos WS. São classes que definem um conjunto de constantes necessárias para o consumo dos WS e um conjunto de métodos públicos que permitem simplificar a invocação dos WS.

Na parte superior do diagrama de classe é possível observar que a aplicação é constituída por três *activities*: "*LoginActivity*", *"FunctionalitiesActivity"* e "*StateTabActivity*". Tal como descrito no diagrama de classes da aplicação do pré-piloto, também neste cenário existe uma classe ("*DataApp*") que permite a partilha de dados entre as várias *activities* da aplicação. Contudo, nesta situação apenas é necessário partilhar um objecto do tipo "*User*".

A *activity*"*LoginActivity*" contém uma classe interna denominada de "*LoginTask*". Esta é uma subclasse da classe *AsyncTask* (consultar a definição no ponto 6 do Anexo 4) que permite realizar operações longas (por exemplo, acesso à rede) em *background* e publicar os resultados no UI. Esta classe é então utilizada para realizar a chamada ao WS que permite a autenticação do utilizador no sistema.

A obtenção da informação do sistema B-Live Wireless é efectuada na "*StateTabActivity*". É obtida informação relativa aos sectores (classe "*Sector*"), divisões (classe "*Room*") e dispositivos (classe "*Device*") presentes no sistema. Por isso, esta *activity* contém um *ArrayList* de objectos para cada uma destas estruturas. As classes "*responseBool*"e "*responseFloat*" são tipos de objectos que reflectem o estado dos dispositivos. Por exemplo, se o dispositivo for uma porta (só pode ter dois estados: aberto ou fechado) e se o atributo "*result*" do objecto "*responsebool*"for *true*, significa que a porta se encontra aberta. Por outro lado, se o dispositivo for, por exemplo, um sensor de temperatura (em que o estado corresponde a um valor numérico), o valor da temperatura é armazenado no atributo "*result*" do objecto "*responsefloat*". O atributo "*isSuccess*" comum às duas estruturas indica se o encontra no atributo "*result*" é válido ou não.

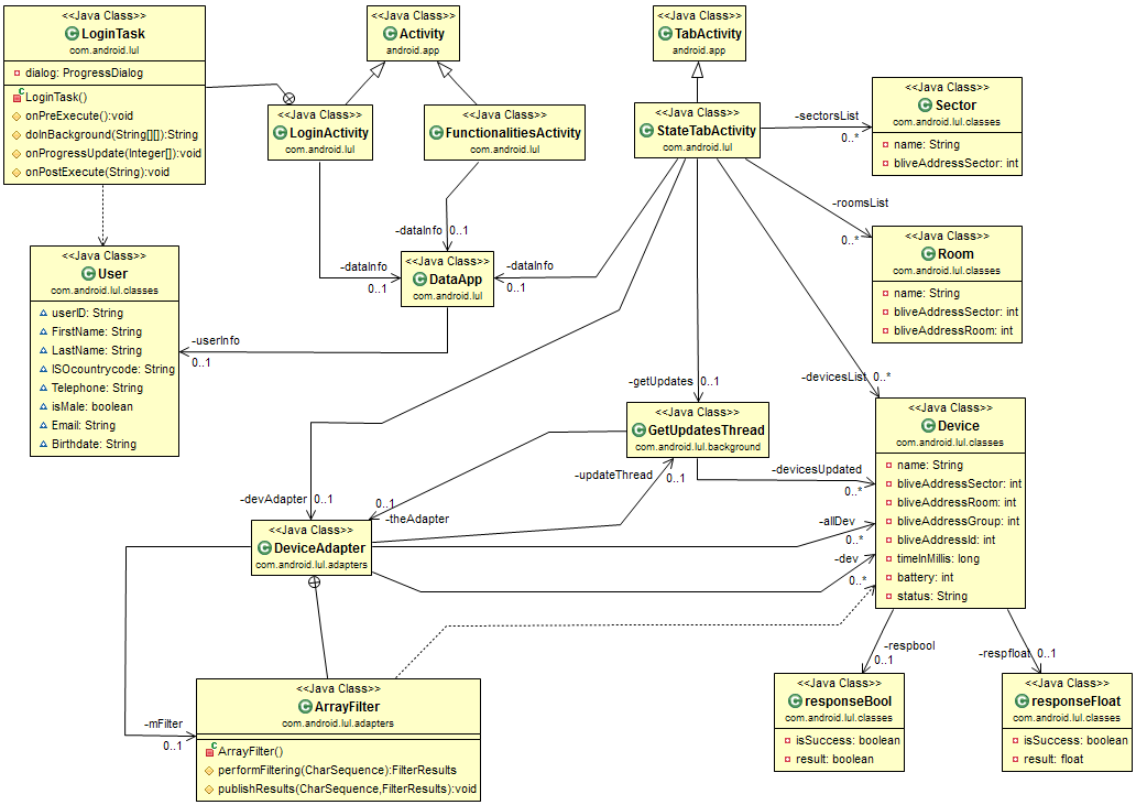


Figura 53 - *Class Diagram* para a aplicação do piloto

A classe "*GetUpdatesThread*" implementa a interface *Runnable*, o que permite a uma instância desta classe ser executada na sua própria *thread*. Esta classe é responsável por invocar periodicamente o WS que verifica a existência de dispositivos actualizados. Relativamente às classes "*DeviceAdapter*" e "*ArrayFilter*", têm as mesmas funcionalidades que na aplicação do pré-piloto, isto é, manipular a informação dos dispositivos que são apresentados na lista e filtrar os dispositivos de acordo com a preferência do utilizador.

Os *Web Services* desenvolvidos e disponibilizados que permitem a obtenção de informação do sistema B-Live podem ser consumidos recorrendo ao protocolo de SOAP (*Simple Object Access Protocol*). Este protocolo é descrito mais pormenorizadamente no Anexo 7 deste documento. A plataforma Android não suporta de forma nativa o consumo de WS SOAP. Contudo, existem APIs desenvolvidas por terceiros que facilitam a implementação deste protocolo na plataforma Android. A mais popular é a kSOAP2 [66], tendo sido esta a solução escolhida para a aplicação.

Os principais serviços disponibilizados que foram utilizados no contexto da aplicação são apresentados na Tabela 12. Adicionalmente, é apresentada uma descrição do serviço bem como alguns dos métodos suportados pelo serviço (são omitidos os parâmetros de entrada e de saída).

Tabela 12 – *Web Services* consumidos na aplicação do piloto

Web Service	Descrição
User Service	Manipulação da informação relativa a um utilizador do sistema B-Live. Métodos: *login()*, *logout()*, *getUserInfo()*, *setUserInfo()*.
Sector Service	Manipulação da informação relativa aos sectores. Métodos: *getAllSectors()*, *setSectorName*, *deleteSector()*.
Room Service	Manipulação da informação relativa às divisões. Métodos: *getAllRooms()*, *getRoomName()*, *getRoomsFromSector()*.
Device Service	Manipulação da informação relativa aos dispositivos. Métodos: *getAllDevices()*, *getAllDevicesFromRoom()*, *checkDeviceUpdates()*, *getLastUpdate()*.
Lamp Service	Manipulação da informação relativa aos dispositivos do tipo lâmpada. Métodos: *getStatus()*, *isTurnOn()*, *turnOn()*, *turnOff()*.
Outlet Service	Manipulação da informação relativa aos dispositivos do tipo tomada. Métodos: *getStatus()*, *isTurnOn()*, *turnOn()*, *turnOff()*.
Door Service	Manipulação da informação relativa aos dispositivos do tipo porta/janela. Métodos: *getStatus()*, *isOpen()*, *open()*, *close()*.
Distance Sensing Service	Manipulação da informação relativa aos sensores de distância. Métodos: *getDistance()*, *getBaseUnits()*.
Temperature Sensing Service	Manipulação da informação relativa aos sensores de temperatura. Métodos: *getTemperature()*, *getBaseUnits()*.
Movement Detection Service	Manipulação da informação relativa aos sensores de movimento. Métodos: *getStatus()*, *isMoving()*.

Como se pode observar na tabela, foi desenvolvido um conjunto de WS que têm em consideração as características dos dispositivos (não foram mencionados todos o WS), sendo da responsabilidade da AC a invocação apropriada de um determinado WS.

Na Figura 54 está representado o *sequence diagram* da interacção entre a aplicação e o servidor B-Live. Como se pode verificar, todas mensagens enviadas por parte da AC correspondem a mensagens síncronas (bloqueantes).

Ao ser pressionado o botão de "Iniciar Sessão" no *layout* de autenticação, é invocado o método de "*login()*" do *User Service*, em que os parâmetros de entrada correspondem ao nome de utilizador e palavra-chave inseridos pelo utilizador. A resposta recebida contém o *userID*[32] atribuído pelo servidor no caso de a autenticação ter sido efectuada com sucesso, ou devolve *null* no caso contrário. Após a autenticação do utilizador no sistema, é efectuado um pedido no sentido de obter a informação relativa ao utilizador através do método "*getUserInfo()*". A resposta a este método é utilizada para inicializar um objecto do tipo *User*.

Seguidamente, já no contexto da funcionalidade de "Estado", são invocados os três métodos que permitem inicializar as estruturas de dados relativas à descrição do sistema B-Live: "*getAllSectors()*", "*getAllRooms()*" e "*getAllDevices()*". Após a obtenção das componentes comuns a todos os dispositivos (nome, bateria, última actualização, etc.) através do método "*getAllDevices*", a lista dos dispositivos é percorrida e são invocados individualmente os métodos específicos de cada tipo de dispositivo para receber a informação relativa ao estado do dispositivo. Por exemplo, para um dispositivo do tipo porta, são invocados os métodos "*isOpen()*" e "*getStatus()*"[33] do *Door Service*. Estas interacções são simbolizadas no diagrama pelo método geral "*getInfoAllDevices()*".

Como foi mencionado, após a obtenção desta informação é criada uma *thread* que verifica periodicamente a existência de alterações de estado dos dispositivos. O método apropriado para este efeito é o "*checkDevicesUpdates()*" do *Device Service*. O período estabelecido para a invocação deste serviço é de 10 segundos, ou seja, de 10 em 10 segundos é verificado se existem alterações no estado dos dispositivos. Este método devolve a lista de dispositivo actualizados e se esta lista não estiver vazia, voltam a ser invocados os métodos específicos de cada tipo de dispositivo como foi referido para o método "*getInfoAllDevices*", embora apenas para os dispositivos actualizados ("*getInfoUpdatedDevices*").

Quando um botão de actuação é pressionado, o método associado à acção do dispositivo é invocado imediatamente. Por isso, no exemplo da porta, se esta estiver fechada, é invocado o método "*open()*" do *Door Service*. Este tipo de método é representado no diagrama pelo método geral "*changeDeviceState()*". No sentido de fazer reflectir o mais rapidamente possível a alteração de estado no UI quando um botão de actuação é pressionado, o método "*checkDevicesUpdates()*" é invocado logo de seguida.

Quando a opção de terminar sessão é pressionada, o método "*logout()*" do *User Service* é invocado e após a recepção da resposta, o *layout* que surge é o da autenticação.

[32] *String* que identifica o utilizador. Este parâmetro é utilizado em todos os métodos de invocação dos WS, pois desta forma o servidor pode devolver informação de acordo com o perfil do utilizador.
[33] Retorna uma *String* com o estado da do dispositivo. No caso da porta retornaria "Aberto" ou "Fechado".

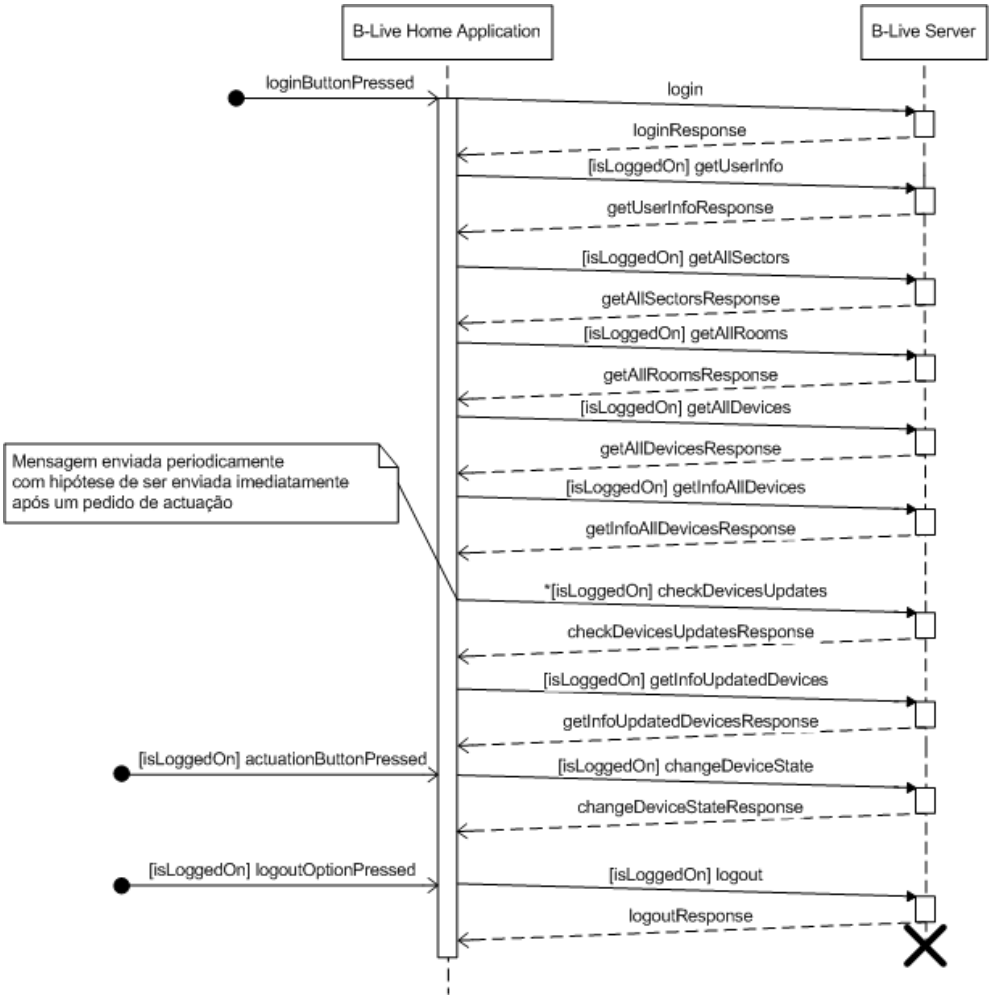


Figura 54 - *Sequence Diagram* da comunicação entre o servidor e a aplicação do piloto

5. Conclusão

A evolução do sector da computação móvel possibilita, cada vez mais, o acesso a qualquer tipo de informação, a qualquer hora e em qualquer lugar. Torna-se assim possível tirar partido destas tecnologias para desenvolver novos produtos (*software* e *hardware*) que ofereçam melhores condições de vida e de bem-estar aos utilizadores, em todo o tipo de ambientes. Um dos projectos que recorre a estas tecnologias é o projecto *Living Usability Lab* e é neste âmbito que esta dissertação se enquadra. Este trabalho consiste no desenvolvimento de uma aplicação móvel, recorrendo à plataforma Android, capaz de ser executada em *smartphones* e *tablets*, que permita a interacção com um sistema de domótica habitacional (B-Live Wireless) para monitorizar e controlar os dispositivos existentes no sistema.

A aplicação desenvolvida é designada por B-Live Home, tendo sido desenvolvidas duas versões, com algumas diferenças, principalmente no modo de comunicação com o servidor do sistema B-Live Wireless. No caso da aplicação para o sistema instalado no Departamento de Investigação, Desenvolvimento e Inovação da Micro I/O, recorreu-se a um protocolo proprietário implementado sobre *sockets* TCP para realizar a comunicação com o servidor. Na aplicação desenvolvida para o piloto da ESSUA, a comunicação com o servidor é baseada em *Web Services*. Foram também realizadas modificações ao nível do aspecto gráfico da interface, por exemplo, ao nível do *set* de cores utilizado, e ainda ao nível da oferta de mais *feedback* visual quando o utilizador interage com a interface.

A utilização de dois métodos distintos para comunicação com o servidor do sistema B-Live Wireless permitiu identificar alguns aspectos que podem ser relevantes na escolha do método mais adequado para a integração da aplicação com o sistema B-Live. De seguida são apresentadas algumas considerações neste contexto.

- Os *Web Services* recorrem ao XML para transferência de mensagens e ao protocolo SOAP que permite a transferência de objectos;
- Ao utilizar *sockets* TCP é necessário ter em consideração o formato definido para as tramas a serem enviadas e recebidas. A composição e decomposição das mensagens é realizada a mais baixo nível (ao nível do *byte*).
- O facto de os *Web Services* recorrerem ao HTTP (camada de aplicação), implica um acréscimo de *overhead*[34] em relação à comunicação por *socket* TCP. Contudo, neste contexto em que pode ser necessário receber muita informação (relativa a todos os dispositivos, divisões, sectores), esse acréscimo não é muito significativo;
- Por outro lado, como o servidor não pode enviar a ocorrência de um evento para o dispositivo móvel imediatamente após a sua detecção (tem de ser a aplicação cliente a invocar o método do WS), pode provocar algum *delay* entre o instante em que ocorre o evento e o instante em que é representada essa alteração no UI.

[34] Mais bytes a serem enviados pela rede devido aos cabeçalhos e ao formato XML, o que gera mais tráfego.

Estas considerações permitem concluir que cada uma das soluções tem as suas vantagens e desvantagens. Não obstante, os *Web Services* parecem ser a solução mais indicada para a implementação.

5.1 Trabalho futuro

Existem vários aspectos que devem ser melhorados na aplicação, destacando-se:

- Adição de novas funcionalidades, como a capacidade para ver o histórico de eventos ocorridos na rede, capacidade para gerir a conta do utilizador, configurar o nome de um dispositivo/divisão/sector;
- Manipulação em *runtime* de falhas de comunicação com o servidor;
- Suporte para outras línguas como o Inglês e o Francês;

Futuramente deverá ser realizada a validação da interface, com o foco da avaliação no UX da aplicação.

Bibliografia

1. Jidenma, N. *What will define the next phase of mobile advertising?* [cited 2012 May]; Available from: http://nl.thenextweb.com/2011/05/22/what-will-define-the-next-phase-of-mobile-advertising/.
2. Nielsvanhelmont. *Mobile services.* [cited 2012 May]; Available from: http://infpower.wordpress.com/2011/03/18/mobile-services/.
3. Initiative, A.H.R. *Home Page.* [cited 2012 June]; Available from: http://awarehome.imtc.gatech.edu/.
4. Alliance, m. *Home Page.* [cited 2012 June]; Available from: http://www.mhealthalliance.org/.
5. Microsoft. *Living Usability Lab.* [cited 2011 October]; Available from: http://www.microsoft.com/pt-pt/mldc/lul/pt/default.aspx.
6. Pandhi, D. *How to Create the Best User Experience for Your Application.* 2006 [cited 2011 December]; Available from: http://msdn.microsoft.com/en-us/library/aa468595.aspx.
7. Nielsen, J. *Usability 101: Introduction to Usability.* [cited 2011 October]; Available from: http://www.useit.com/alertbox/20030825.html.
8. Usernomics. *Home Page.* [cited 2011 November]; Available from: http://www.usernomics.com/.
9. Raskin, J., *Intuitive equals familiar.* Association for Computing Machinery. Communications of the ACM, 1994. **37**(9): p. 17.
10. Firtman, M. *UI Guidelines for mobile and tablet web app design.* 2010; Available from: http://www.mobilexweb.com/blog/ui-guidelines-mobile-tablet-design.
11. Microsoft. *Windows Phone Development.* [cited 2012 May]; Available from: http://msdn.microsoft.com/en-us/library/ff402535(v=vs.92).
12. Apple. *iOS Dev Center.* [cited 2012 May]; Available from: https://developer.apple.com/devcenter/ios/index.action.
13. Google. *Android Developers.* [cited 2012 May]; Available from: http://developer.android.com/index.html.
14. Microsoft. *Home Page.* [cited 2012 May]; Available from: http://www.microsoft.com.
15. Koh, D. *Q&A: Microsoft on Windows Phone 7 Series.* 2010 [cited 2012 May]; Available from: http://asia.cnet.com/qanda-microsoft-on-windows-phone-7-series-62061278.htm.
16. Nokia. *Home Page.* [cited 2012 May]; Available from: http://www.nokia.com/in-en/.
17. Brodkin, J. *Microsoft/Nokia partnership paying off as developers eye Windows Phone.* 2011; Available from: http://arstechnica.com/business/2011/11/microsoftnokia-partnership-paying-off-as-developers-eye-windows-phone/.
18. Kumar, S. *After C# , now its Visual Basic's turn for Windows Phone 7.* 2010 [cited 2012 May]; Available from: http://www.qinktage.com/2010/09/after-c-now-its-visual-basics-turn-for-windows-phone-7/.
19. Microsoft. *Windows Phone Marketplace.* [cited 2012 May]; Available from: http://www.windowsphone.com/en-US/marketplace.
20. Apple. *Home Page.* [cited 2012 May]; Available from: http://www.apple.com/.
21. Cohen, J. *What Languages Does iPhone Use?* 2011 [cited 2012 May]; Available from: http://www.ehow.com/info_8332185_languages-iphone-use.html.
22. Apple. *Apple Store.* [cited 2012 May]; Available from: http://store.apple.com/us.
23. Alliance, O.H. *Home Page.* [cited 2012 November]; Available from: http://www.openhandsetalliance.com/.
24. Project, A.O.S. *Philosophy and Goals.* [cited 2012 May]; Available from: http://source.android.com/about/philosophy.html.

25. Paul, R. *Android goes beyond Java, gains native C/C++ dev kit*. [cited 2012 May]; Available from: http://arstechnica.com/gadgets/2009/06/android-goes-beyond-java-gains-native-cc-dev-kit/.
26. Eclipse. *Home Page*. [cited 2011 December]; Available from: http://www.eclipse.org/.
27. PhoneGap. *Home Page*. [cited 2012 May]; Available from: http://phonegap.com/.
28. Adobe. *Home Page*. [cited 2012 May]; Available from: http://www.adobe.com/.
29. Blackberry. *Home Page*. [cited 2012 May]; Available from: http://us.blackberry.com/.
30. WebOS. *Home Page*. [cited 2012 May]; Available from: https://developer.palm.com/.
31. Bada. *Home Page*. [cited 2012 May]; Available from: http://www.bada.com/.
32. RhoMobile. *Home Page*. [cited 2012 May]; Available from: http://www.rhomobile.com/.
33. MoSync. *Home Page*. [cited 2012 May]; Available from: http://www.mosync.com/.
34. WORTHAM, J. and N. WINGFIELD *Microsoft Is Writing Checks to Fill Out Its App Store*. The New York Times, 2012.
35. Gartner, I., *Gartner Says Worldwide Smartphone Sales Soared in Fourth Quarter of 2011 With 47 Percent Growth*.
36. Douangboupha, P., *Smart Phone Platform Comparison*, in *A Comparative Analysis*, R2integrated, Editor. p. 9.
37. Mifsud, J. *Official Usability, User Experience & User Interface Guidelines From Companies*. [cited 2011 October]; Available from: http://usabilitygeek.com/official-usability-user-experience-user-interface-guidelines-from-companies/.
38. Botella, P., et al., *ISO/IEC 9126 in practice: what do we need to know?* , in *Procs. First Software Measurement European Forum*2004: Rome.
39. Chua, B.B. and L.E. Dyson, *Applying the ISO 9126 model to the evaluation of an elearning system*, in *Proc. of ASCILITE*2004.
40. ISO/IEC, *9241-11 Ergonomic requirements for office work with visual display terminals (VDT)s*, in *Part 11 Guidance on usability*1998.
41. ISO/IEC, *11581-1 Information technology - User system interfaces and symbols - Icons symbols and functions*, in *Part 1 Icons -General*2000.
42. Techscribe. *The development of ISO/IEC 18019:2004*. [cited 2011 November]; Available from: http://www.techscribe.co.uk/techw/iso-18019.htm.
43. Nielsen, J. *Ten Usability Heuristics*. [cited 2011 November]; Available from: http://www.useit.com/papers/heuristic/heuristic_list.html.
44. Dmytro. *UX Best Practices for Mobile Development*. [cited 2012 February]; Available from: http://blog.sphereinc.com/2012/01/ux-best-practices-for-mobile-applications/.
45. Iniciative, O.S. *Home Page*. [cited 2011 December]; Available from: http://www.opensource.org/.
46. Iniciative, O.S. *Home Page*. [cited 2011 November]; Available from: http://www.opensource.org/.
47. Foundation, T.A.S. *Apache License, Version 2.0*. 2004 [cited 2011 December]; Available from: http://www.apache.org/licenses/LICENSE-2.0.
48. System, G.O. *GNU General Public License, version 2*. [cited 2011 December]; Available from: http://www.gnu.org/licenses/gpl-2.0.html.
49. System, G.O. *GNU General Public License, version 3*. [cited 2011 December]; Available from: http://www.gnu.org/licenses/gpl-3.0.html.
50. McGregor, C. *Interview with Richard M. Stallman*. 2008 [cited 2011 December]; Available from: http://www.freesoftwaremagazine.com/articles/interview_with_richard_stallman.
51. Khason, T. *Open Source licenses comparison table*. [cited 2011 December]; Available from: http://khason.net/blog/open-source-licenses-comparison-table/.
52. Baptista, V.J.D., *B-Live Wireless: A Real-Time Protocol For Home Automation*, in *Departamento de Electrónica, Telecomunicações e Informática*2011, Universidade de Aveiro. p. 132.
53. Micro I/O, Serviços de Electrónica, Lda., *Pré-piloto B-Live Wireless*. Relatório interno, 29 de outubro de 2011.
54. Micro I/O, Serviços de Electrónica, Lda., *Piloto B-Live Wireless*. Relatório interno, 2 de julho de 2012.

55. Foundation, T.A.S. *Welcome to Apache Axis2/Java*. [cited 2012 May]; Available from: http://axis.apache.org/axis2/java/core/.
56. Foundation, T.A.S. *Apache Tomcat*. [cited 2012 May]; Available from: http://tomcat.apache.org/.
57. Google. *Platform Versions*. [cited 2012 May]; Available from: http://developer.android.com/resources/dashboard/platform-versions.html.
58. Samsung. *Galaxy Gio*. [cited 2012 July]; Available from: http://www.samsung.com/pt/consumer/mobile-phone/smartphones/android/GT-S5660DSATCL.
59. Samsung. *Galaxy Note*. [cited 2012 July]; Available from: http://www.samsung.com/pt/consumer/mobile-phone/note/note/GT-N7000ZBAOPT.
60. Asus. *Eee Pad Transformer Prime TF201*. [cited 2012 July]; Available from: http://pt.asus.com/Eee/Eee_Pad/Eee_Pad_Transformer_Prime_TF201/.
61. Studios, B. *Home Page*. [cited 2011 December]; Available from: http://www.balsamiq.com/.
62. *Color Scheme Designer*. [cited 2012 March]; Available from: http://colorschemedesigner.com.
63. Adobe. *Adobe Illustrator*. [cited 2012 February]; Available from: http://www.adobe.com/products/illustrator.html.
64. Adobe. *Adobe Photoshop*. [cited 2012 February]; Available from: http://www.adobe.com/products/photoshop.html.
65. Inc., A. *Introduction to the Diagrams of UML 2.0*. [cited 2012 June]; Available from: http://www.agilemodeling.com/essays/umlDiagrams.htm.
66. Moser, M. *ksoap2-android*. [cited 2012 April]; Available from: http://code.google.com/p/ksoap2-android/.
67. Developer, E. *Android – Multithreading in a UI environment*. [cited 2012 June]; Available from: http://www.aviyehuda.com/2010/12/android-multithreading-in-a-ui-environment/.
68. *Android Asset Studio*. [cited 2012 February]; Available from: http://android-ui-utils.googlecode.com/hg/asset-studio/dist/index.html.
69. Cunha, D. *Web Services, SOAP e Aplicações Web*. [cited 2012 June]; Available from: http://devedge-temp.mozilla.org/viewsource/2002/soap-overview/index_pt_br.html.
70. Rathnayake, S. *Android Web Service Access Tutorial*. [cited 2012 July]; Available from: http://sarangasl.blogspot.pt/2011/10/android-web-service-access-tutorial.html.

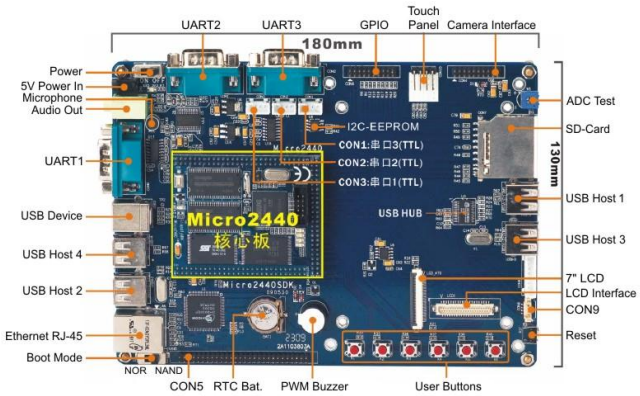
Anexos

Anexo 1: Plataformas de *hardware* para desenvolvimento Android

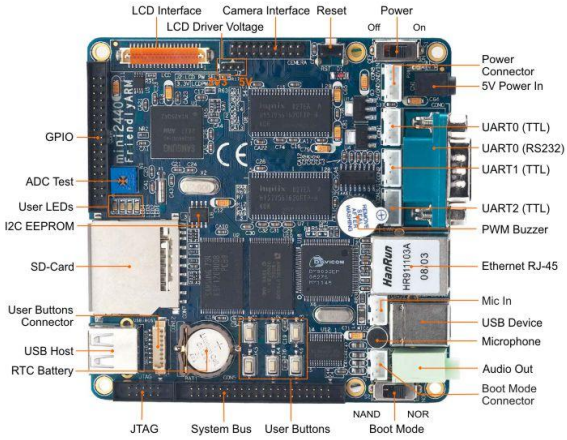
Neste anexo são apresentados algumas plataformas de desenvolvimento de *hardware* existentes no mercado, nas quais é possível executar o sistema operativo Android.

- **MICRO2440:** http://www.friendlyarm.net/products/micro2440

Esta plataforma de desenvolvimento é baseada no microcontrolador da Samsung S3C2440 (ARM9), trata-se de uma *board* desenvolvida pela FriendlyARM cujo sistema operativo instalado por defeito é o Linux, com documentos muito bem detalhados acerca das configurações necessárias para alterar o SO a ser utilizado. É uma placa com 180x130mm, com várias interfaces, portas e conectores aos quais se podem ligar diversos componentes.



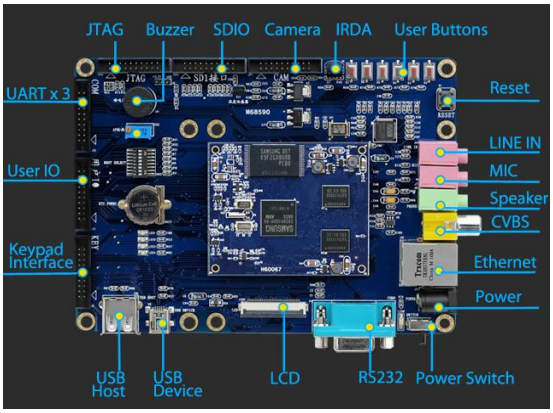
- **MINI2440:**http://www.friendlyarm.net/products/mini2440



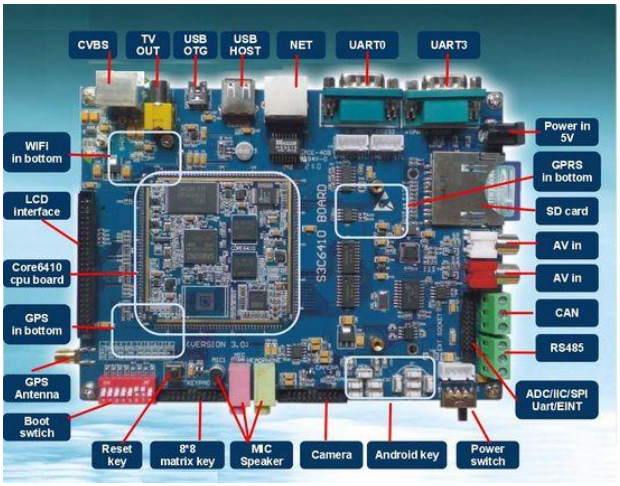
Este é um modelo bastante semelhante ao anterior, também desenvolvida pela FriendlyARM, é uma das placas mais comum no mercado. Também usufrui das possibilidades oferecidas pelo microcontrolador da Samsung S3C2440 (ARM9). A nível de *hardware* difere do modelo anterior essencialmente no número de interfaces, por exemplo, enquanto a anterior tem duas portas RS232 e quatro USB, esta tem uma e duas respectivamente.

- **OK6410:** http://www.arm9board.net/(X(1)S(23p04345tkojr545fms2gr55))/sel/prddetail.aspx?id=348&pid=200

A OK6410 é uma SBC (*single board computer*) baseada no microcontrolador da Samsung S3C6410 (ARM11). Possui uma capacidade de processamento de vídeo bastante poderosa. Juntamente com a placa é fornecida informação acerca da plataforma, incluindo drivers para todas as componentes presentes, para Linux, Windows CE e Android.



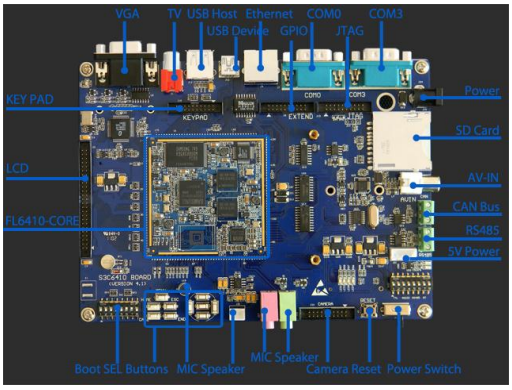
- **REAL6410:**http://www.real6410.com/realarm/Real6410%20board.html



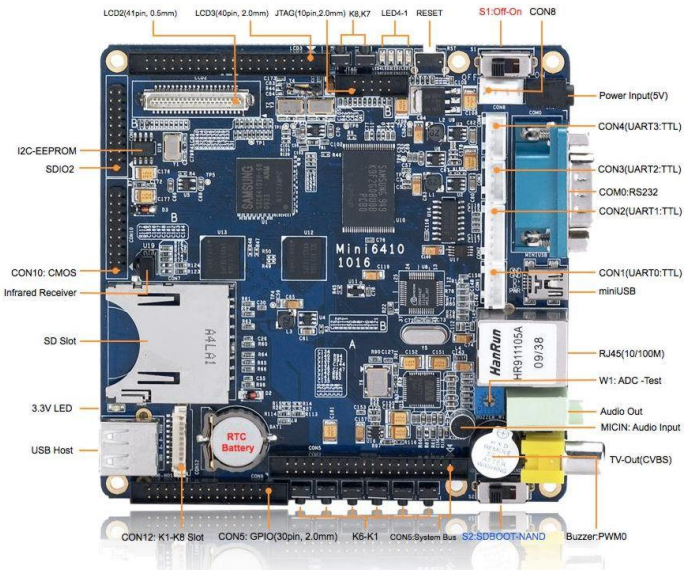
Baseado no µC da Samsung S3C6410 (ARM11) e produzido pela CoreWind Technology, esta plataforma oferece várias funções como RS232, USB, Ethernet, Wifi, GPS, entre outros, o que torna esta SBC numa opção válida no desenvolvimento de soluções baseadas em processadores ARM. Também esta plataforma oferece suporte aos diversos sistemas operativos utilizados em sistemas embutidos.

- **FL6410:** http://www.arm9board.net/sel/prddetail.aspx?id=363&pid=200

Tal como o modelo OK6410, este é produzido e comercializado pela Witech Co., Ltd. e é baseado também no µC da Samsung S3C6410 (ARM11). Apesar de ser um modelo parecido com o OK6410, existem algumas diferenças entre eles, a começar pelo preço, o que indica diferenças também a nível de *hardware*. Algumas destas diferenças serão postas em evidência mais a frente.



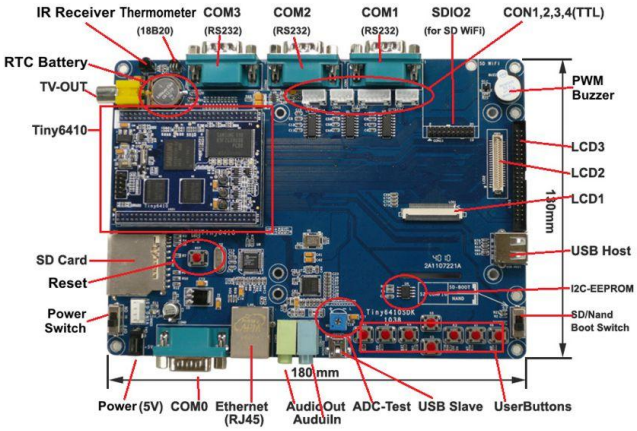
- **MINI6410:**http://www.friendlyarm.net/products/mini6410



A *board* de alto desempenho MINI6410 é desenvolvida pela FriendlyARM, e baseado no µC da Samsung S3C6410 (ARM11). Com algumas características como 256MB SDRAM, 1GB NAND flash, RTC, USB e RS232 entre muitas outras, e com suporte para vários sistemas operativos como Linux Embedded, Android, Windows CE, tornam esta placa numa referência sólida como plataforma de desenvolvimento de novas soluções. Muito usada em sistemas de multimédia e comunicação.

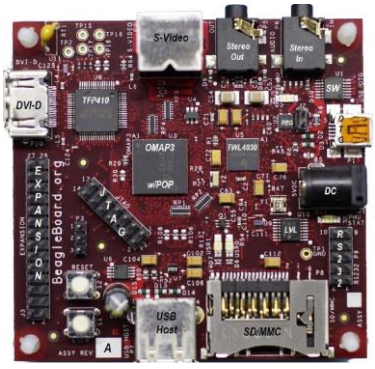
- **TINY6410:**http://www.friendlyarm.net/products/tiny6410

Outra das placas produzidas pela FriendlyARM é a TINY6410, que tal como a anterior é baseada no µC da Samsung S3C6410 (ARM11). Tal como acontece nos modelos MICRO2440 e MINI2440, o que difere este modelo do anterior a nível de *hardware* é basicamente o número de interfaces num e noutro, mas também, o TINY6410 ter integrado um sensor de temperatura e o MINI6410 não ter. Outra das diferenças é o sistema de entrada de áudio: o MINI6410 tem microfone incorporado e o TINY6410 usar conectores de 3,5mm.

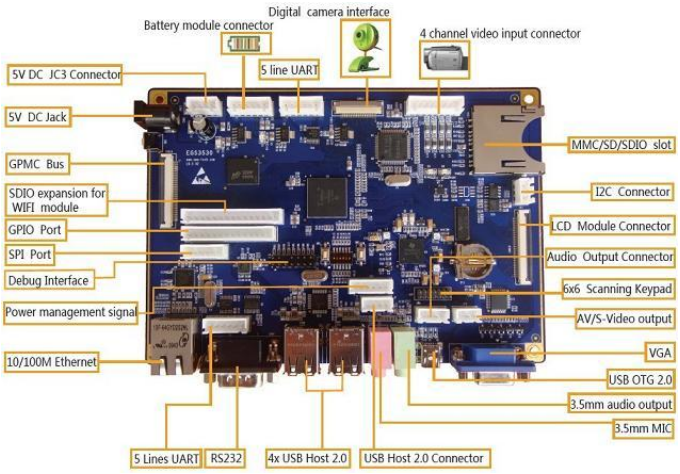


- **BeagleBoard-xM:** http://beagleboard.org/hardware

A plataforma de desenvolvimento BeagleBoard, de baixo custo e consumo é desenvolvida pela Texas Instruments. Para além de demonstrar as potencialidades do processador OMAP3530 (ARM Cortex-A8), esta placa tem o intuito de poder ser utilizado na área da educação (mas não só) de forma a exemplificar o uso de *hardware* e *softwareopen source*. Para além das muitas características que esta *board* apresenta, existe a possibilidade de torná-la completamente portátil, pois há um módulo que incorpora uma bateria capaz de alimentar o circuito.



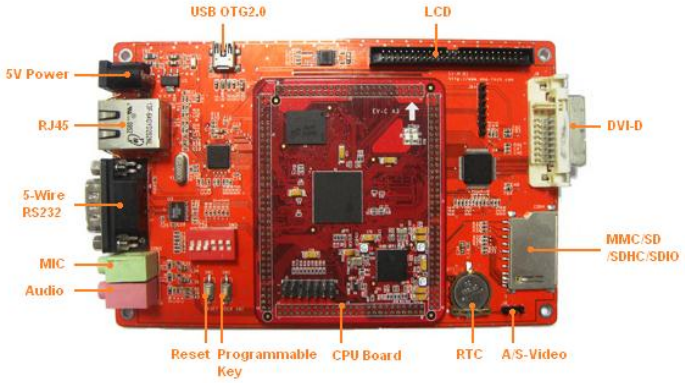
- **SBC3530:**http://www.armdesigner.com/SBC3530.html



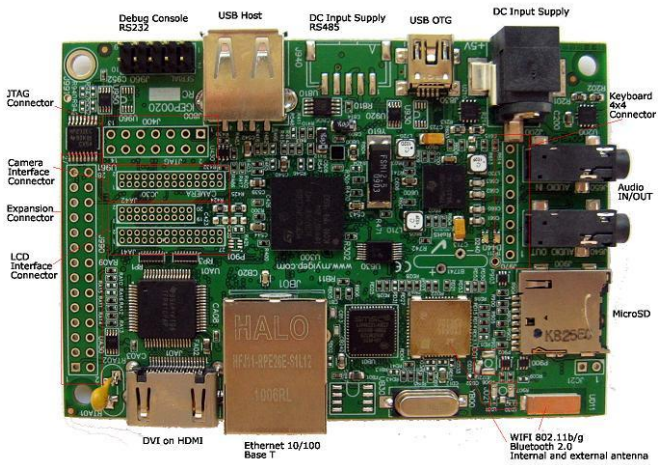
A SBC3530 é uma *board* de alta *performace* e baixo consumo, baseado num processador da TI (Texas Instruments), o OMAP3530 (ARM Cortex-A8). É produzida pela GUANGZHOU EMBEDDED MACHINE TECHNOLOGY CO. Com resoluções até 1920x1080, suporta várias saídas para display, como VGA, LCD, S-Video e AV. Suporta Linux, Windows CE e Android. É bastante usado nas áreas de equipamento médico, GPS…

- **KIT OMAP3530:**http://www.armdesigner.com/KIT%20OMAP3530.html

Esta plataforma, também ela baseada no processador OMAP3530 (ARM Cortex-A8), o que torna possível uma performance 4 vezes superior que um processador baseado em ARM9. É desenvolvido pela Boardcon Embedded Design. A *board* fornece todo o material de suporte ao sistema operativo Linux, embora este também seja capaz de suportar Windows CE e Android. Tem suporte também para vários periféricos adicionais que não se encontram integrados no KIT por defeito.



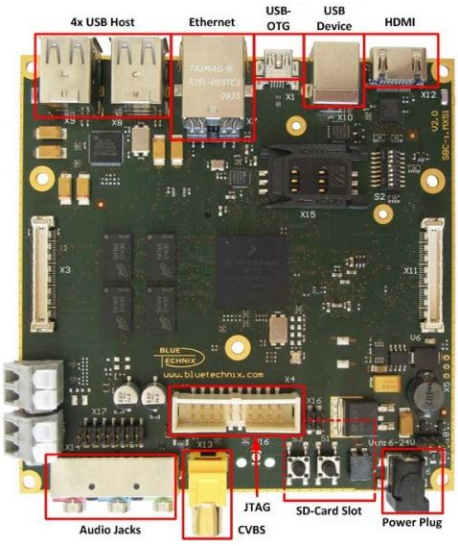
- **IGEPv2:**http://www.igep-platform.com/index.php?option=com_content&view=article&id=46&Itemid=55



Com tamanho pouco maior que um cartão de crédito, esta é uma *board* com muito potencial no desenvolvimento de sistemas embutidos e não só. Produzido pela ISEE, com o OMAP3530 (ARM Cortex A-8) integrado, esta plataforma de desenvolvimento oferece um conjunto de funcionalidades que nem todos suportam. No entanto, para se retirar os maiores benefícios na utilização desta placa, convém adicionar a esta a placa de expansão (IGEPv2 Expansion *board*), pois fornece vários outros recursos como interface para LCD, modem GSM/GPRS, CAN *bus interface*…

- **SBC-i.MX51:** http://pt.farnell.com/bluetechnix/sbc-i-mx51/board-sbc-i-mx51-w-cortex-a8/dp/1825712

Trata-se de uma plataforma móvel produzida pela Blue Thechnix, de alto desempenho, baseado em ARM Cortex-A8. Possui unidade de processamento de imagem e unidade de processamento de vídeo. As suas capacidades de memória e as diversas interfaces que apresenta tornam esta SBC numa boa solução para sistemas embutidos aplicados em ambientes industriais e sistemas de controlo por exemplo.



Anexo 2: Lista das componentes das normas apresentadas

A lista abaixo apresenta as componentes da norma ISO 9241:

- *Part 1: General lntroduction*
- *Part 2: Guidance on task requirements*
- *Part 3: Visual display requirements*
- *Part 4: Keyboard requirements*
- *Part 5: Workstation layout and postural requirements*
- *Part 6: Environmental requirements*
- *Part 7: Requirements for display with reflections*
- *Part 8: Requirements for displayed colours*
- *Part 9: Requirements for non-keyboard input devices*
- *Part 10: Dialogue principles*
- *Part 11: Guidance on usability*
- *Part 12: Presentation of information*
- *Part 13: User guidance*
- *Part 14: Menu dialogues*
- *Part 15: Command dialogues*
- *Part 16: Direct manipulation dialogues*
- *Part 17: Form-filling dialogues*

A lista abaixo apresenta as componentes da norma ISO/IEC 11581:

- *Part 1: Icons – General*
- *Part 2: Object icons*
- *Part 3: Pointer icons*
- *Part 4: Control icons*
- *Part 5: Tool icons*
- *Part 6: Action icons*

Anexo 3: Instalação do SDK para desenvolvimento Android

Neste anexo é apresentado um guia passo a passo para a instalação do ambiente de desenvolvimento para Android no sistema operativo Windows, baseado no guia disponibilizado pela Google no site de apoio aos desenvolvedores Android. Também é possível instalar este ambiente de desenvolvimento em Ubuntu e em MAC OS X.

- **Instalação do JDK**

Um dos requisitos prévios para a instalação do SDK é que o sistema computacional deve ter instalado o JDK versão 6 ou superior. Por isso, caso este ainda não esteja instalado, é necessário ir ao *site* da Oracle e efectuar o *download* gratuito do JDK. O *site* onde é possível fazer o *download* é o seguinte:

http://www.oracle.com/technetwork/java/javase/downloads/index.html

Depois de efectuar o download do JDK apropriado, é necessário instalá-lo no sistema. Este processo é simples, bastando seguir os passos apresentados ao longo da instalação.

- **Eclipse IDE**

O IDE recomendado pela Google para o desenvolvimento em Android é o Eclipse. A versão a utilizar deve ser "Eclipse Classic" e pode ser obtida de forma gratuita a partir do seguinte *site*:

http://www.eclipse.org/downloads/

O Eclipse não necessita de ser instalado, basta descomprimir o ficheiro recebido para um directório acessível, por exemplo, o *Desktop*. Para inicializar o IDE, basta entrar no directório do Eclipse e executar o ficheiro "eclipse.exe".

- **Instalação do SDK Starter Package**

De seguida é necessário fazer o download do SDK Starter Package e proceder à sua instalação. Este elemento não contém todas as componentes do SDK, apenas contém algumas, sendo as restantes obtidas posteriormente. O download pode ser efectuado através do *link*:

http://developer.android.com/sdk/index.html

É aconselhável o download do ficheiro no formato *.zip. Para proceder à instalação, extrai-se o conteúdo do ficheiro *.zip para um directório acessível. O directório não deve conter espaços ou acentos (por exemplo, C:\). Por fim, dentro da pasta do SDK, executar o "SDK Manager", seleccionar a *chekbox* do *package* "Android SDK Platform-tools" e clicar em *Install 1 package*. Aguardar até ao final da instalação.

- **Instalação do ADT *plugin* no Eclipse**

Este *plugin* permite integrar no Eclipse IDE todas as ferramentas necessárias para o desenvolvimento de aplicações Android. Os passos para a instalação deste *plugin* são apresentados de seguida.

a) Iniciar o Eclipse, no menu seleccionar *Help > Install New Software…*

b) Clicar em *Add*.

c) No parâmetro *Name* colocar "ADT Plugin" e em *Location* colocar https://dl-ssl.google.com/android/eclipse/.

d) Clicar em *OK*. Se neste ponto ocorrer um erro, trocar "https" por "http".

e) Seleccionar a *checkbox* do "Developer Tools" e clicar em *Next*.

f) Na janela apresentada é possível visualizar as componentes a instalar. Clicar em *Next*.

g) Ler e aceitar os termos e clicar em *Finish*.[35]

h) Quando a instalação finalizar, é necessário reiniciar o Eclipse.

- **Configuração do ADT *plugin***

De seguida é necessário proceder à correcta configuração do *plugin* instalado.

a) No Eclipse, seleccionar *Window > Preferences*.

b) Na lista do lado esquerdo, escolher *Android*, aparecendo um aviso relativo às estatísticas da Google no qual se deve clicar em *Proceed*.

c) Em *SDK Location*, clicar em *Browse…* e colocar o *path* do directório onde de encontra o SDK instalado anteriormente (por exemplo, C:\android-sdk-windows).

d) Clicar em *Apply*.

e) Por fim, clicar em *OK*.

- **Adição de plataformas Android**

Para se poder desenvolver aplicações para as diversas versões de Android é necessário ter instalado as *frameworks* adequadas para tal. Neste ponto é apresentado o processo de instalação do *software* necessário para suportar as diversas versões.

a) No Eclipse, seleccionar *Window > Android SDK Manager*.

b) Seleccionar as *checkboxes* das versões que se pretende (por exemplo, 4.0, 3.0, 2.3) e a dos extras.

c) Clicar em *Install…*

d) Na janela seguinte, escolher *Accept All* e clicar em *Install*.

[35] Se durante a instalação aparecer algum aviso relativo a autenticidade ou validade, clicar em *OK*.

e) Esperar pelo fim da instalação, que pode demorar algum tempo. Se durante a instalação aparecer uma janela de autenticação no site "Motodev", clicar em *Cancel*.

f) Reiniciar o Eclipse.

- **Criação de um emulador Android**

O SDK possibilita simular a execução das aplicações num emulador, ou seja, num dispositivo virtual. A criação deste dispositivo virtual é efectuada da seguinte forma:

a) No Eclipse, seleccionar *Window > AVD Manager*.

b) Clicar em *New…*

c) Preencher os campos e clicar em *Create AVD.*

d) O emulador encontra-se criado. Para o inicializar basta clicar em *Start*.

- **Criação de um projecto em Android**

Agora que o ambiente de desenvolvimento se encontra instalado é possível criar um projecto Android. Para a criação do projecto são necessários efectuar os seguintes passos:

a) No Eclipse, seleccionar *File > New > Project…*

b) Na pasta *Android*, escolher *Android Project* e clicar em *Next*.

c) Seleccionar a versão alvo, que deve ser igual à versão escolhida para o emulador criado.

d) Clicar em *Next*.

e) Escolher o nome do *package* (por exemplo, com.android.ui).

f) Clicar em *Finish*.

Anexo 4: Guias de referência e conceitos para desenvolvimento em Android

1. Estrutura de um projecto em Android

A Figura 55 ilustra o ambiente gráfico de desenvolvimento em Android utilizando o Eclipse IDE. Do lado esquerdo da ilustração, é possível verificar a listagem dos projectos contidos na *tab "Package Explorer"*. É neste local que podem ser observados a estrutura/hierarquia de um determinado projecto.

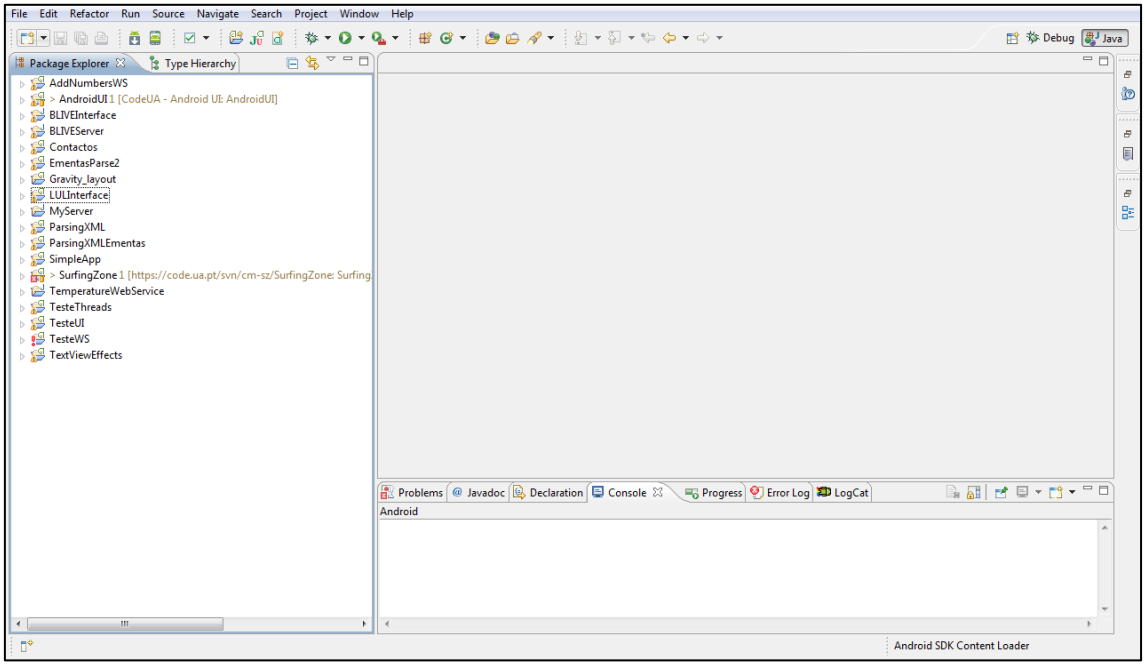


Figura 55 – Painel do ambiente de desenvolvimento Android (Eclipse IDE)

A escolher e abrir um projecto, é apresentado o primeiro nível da hierarquia. Este nível é constituído pelas seguintes pastas/ficheiros[36]:

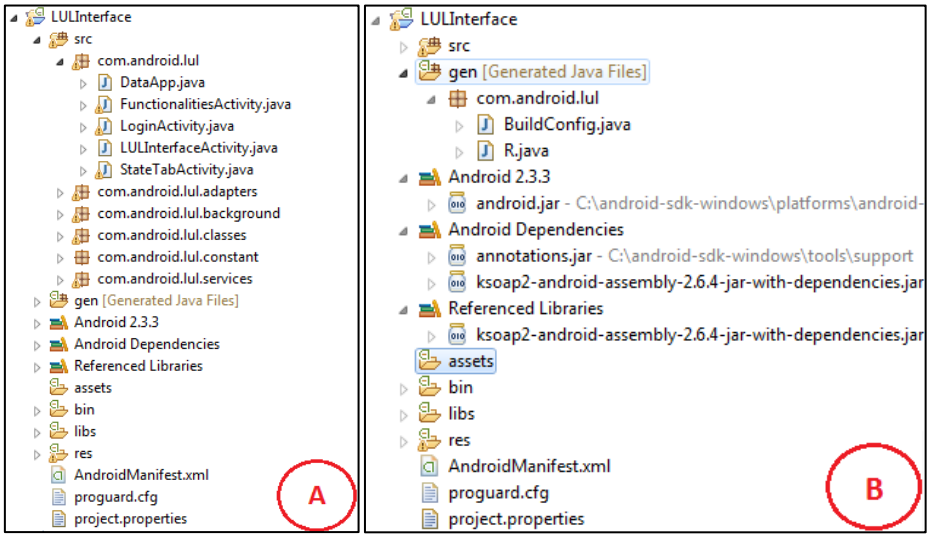
- /src
- /gen
- /Android <*version number*>
- /Android Dependecies
- /Referenced Libraries
- /assets

[36] Nem todas as pastas são criadas aquando da criação do projecto, algumas podem ser adicionadas posteriormente consoante a necessidade

- /bin
- /libs
- /res
- AndroidManifest.xml
- proguard.cfg
- project.properties

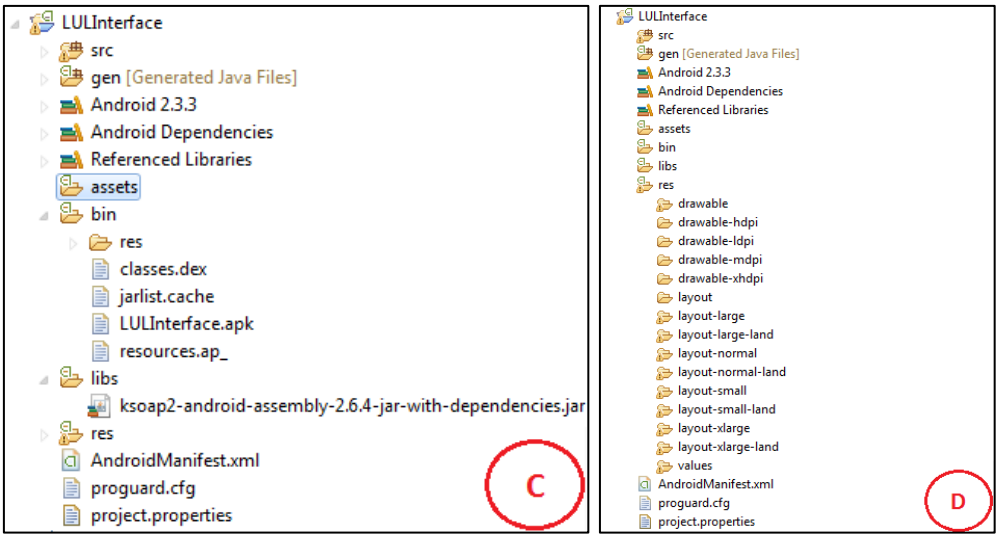
As imagens seguintes visam ilustrar o segundo nível da hierarquia das pastas mencionada acima. As imagens são identificadas pela letra a vermelho presente no canto inferior direito de cada imagem.

Na imagem A, é possível observar o conteúdo da pasta "/src". Esta é a pasta que contém o código Java da aplicação organizado em *packages*. Já na imagem B é apresentado o conteúdo de várias pastas. Na pasta "/gen" encontram-se também ficheiros Java, contudo, estes ficheiros são gerados automaticamente pela plataforma aquando da compilação do projecto, e não devem ser modificados. A função do ficheiro "R.java" é conter as referências para os vários recursos (*resources*) presentes no projecto, como se pode verificar mais abaixo. As pastas "/Android 2.3.3", "/Android Dependecies" e "/Referenced Libraries"dizem respeito às bibliotecas utilizadas no projecto, assim como a pasta "/libs".



Na pasta "/assets" pode conter ficheiros de qualquer tipo para serem usados como *resources* para a aplicação a desenvolver e são guardados no formato *raw* podendo apenas ser acedidos programaticamente. Na imagem C pode-se observar o conteúdo da pasta "/bin" que contem os ficheiros gerados após a compilação, incluindo o ficheiro ".apk" que é o executável da aplicação para sistemas operativos Android. Na imagem D, é apresentado o conteúdo típico de uma pasta "/res". Esta é a pasta dos *resources* incluídos no projecto. Estes *resources* podem ser divididos em várias categorias: *drawable* (imagens[37]), *layouts* (definição do UI), *values* (por exemplo, *strings*, cores, *arrays*, dimensões), entre outros.

[37]Imagens em vários formatos possíveis tais como PNG ou definidos em vocabulário XML.



A pasta "/res" pode e deve ser organizada de forma a providenciar diferentes *resources* baseados nas configurações do dispositivo móvel, através de nomes específicos para as suas subpastas. Depois, em *runtime*, a aplicação utilizará a recurso apropriado de acordo com as configurações actuais do dispositivo. Exemplos de configurações são a linguagem/região definida para o dispositivo, o tamanho do ecrã, a densidade do ecrã, a orientação do dispositivo, o modo de *touch* (dedo, caneta, sem modo de toque). Estas e outras configurações são suportadas recorrendo aos denominados *qualifiers* que compõem o nome das subpastas da pasta "/res". O nome das subpastas devem seguir a seguinte regra:

<resource_name>-<config_qualifier>

É possível utilizar mais do que uma *qualifier*, separados por hífenes, embora respeitando uma certa ordem. Exemplos desta regra:

1. drawable-hdpi
2. layout-small-land
3. values-en

Na subpasta do exemplo 1, devem ser colocados *resources* do tipo imagem para a suportar um ecrã de elevada densidade de pixéis. No exemplo 2, são inseridos os *layouts* para ecrãs de reduzida dimensão e em que a orientação se encontre na vertical. Relativamente ao exemplo 3, são colocados os *resources* a serem usados quando o dispositivo se encontra configurado com a linguagem em inglês.

Por último, ainda na imagem D, de referir a existência do ficheiro "AndroidManifest.xml". Este ficheiro é muito importante e deve constar em todos os projectos Android. Contêm informação relativa à aplicação tais como as *activities* (definição mais à frente neste anexo) presentes na aplicação, serviços, versão do SDK e do projecto, permissões necessárias para a execução da aplicação, entre outros.

2. *User Interface* (*guidelines* para versões 2.3.7 ou inferiores)

Como já foi referido, o Android possui os seus próprios guias de referência para o desenvolvimento de interfaces gráficas e para a interacção das suas aplicações. Estes *guidelines*, estão divididos em quatro categorias: ícones, *widgets*, menus, actividades/tarefas. De seguida, são apresentados os aspectos mais importantes relativos a cada categoria, sendo os exemplos retirados do site de apoio ao desenvolvimento Android [13].

a) Icon design guidelines

Em aplicações Android, existem vários tipos de ícones, classificados da seguinte forma: *launcher, menu, action bar, status, tab, dialog* e *list view.*

Os ***launcher icons*** representam a aplicação. Estes ícones devem possuir as seguintes características:

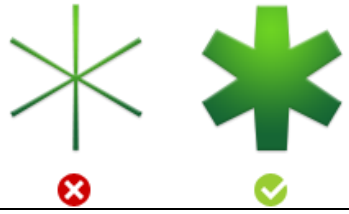
- Únicos e memorizáveis;
- As cores devem combinar com a aplicação;
- Não se deve incluir o nome da aplicação no ícone;
- Boa conjugação com uma ampla variedade de backgrounds;
- Perceptível mesmo em tamanhos pequenos;
- Forma única para mais rápido reconhecimento;
- Não devem ser demasiado largos ou compridos.

Na Tabela 13, são apresentados alguns exemplos do que é considerado correcto e errado. Estes ícones devem estar no formato 32-bit PNGs (Portable Network Graphics) com canal alfa[38] para transparência.

Tabela 13 - Exemplos do que é considerado correcto e incorrecto nos *launcher ícons*

	Não usar ícones demasiado complexos.
	Ícones não devem aparecer cortados.

[38]Define a opacidade de um pixel numa imagem.

	Ícones não devem ser muito finos.
	Ícone deve ser simples mas quando apropriado, deve-se distinguir o ícone com um tratamento visual subtil mas apelativo.

Relativamente às dimensões, estas variam de acordo com a densidade do ecrã e estão indicadas na Tabela 14.

Tabela 14 - Dimensões dos ícones em pixéis

	ldpi (120dpi) *(Low density screen)*	**mdpi** (160dpi) *(Medium density screen)*	**hdpi** (240dpi) *(High density screen)*	**xhdpi** (320dpi) *(Extra-high density screen)*
Tamanho ícone	36x36 px	48x48 px	72x72 px	96x96 px

Os ***menu icons*** são elementos que são mostrados quando o utilizador clica no botão Menu. Encontram-se numa perspectiva frontal e em escala de cinzento. Também nestes ícones é necessário ter em conta a densidade do ecrã, mas também a forma do ícone (mais arredondado – limite azul, mais quadrado – limite laranja). Os posicionamentos e os tamanhos estão apresentados na Tabela 15.

Tabela 15 - Posicionamento e dimensões dos *menu ícons*

(hdpi) *screens:* *Full Asset:* 72x72px *Icon:* 48x48 px *Square Icon:* 44x44 px	
(mdpi) *screens:* *Full Asset:* 48x48 px *Icon:* 32x32 px *Square Icon:* 30x30 px	

(ldpi) ***screens:*** *Full Asset:* 36x36 px *Icon:* 24x24 px *Square Icon:* 22x22 px	

Os ícones com uma forma aproximadamente quadrada, necessitam de ter dimensões menores, para manter uma consistência visual com os outros tipos de ícones.

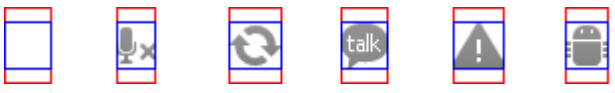
Os ***actions bar icons*** representam itens de acção individual localizados na *Action Bar*. Usados para versões do Android 3.0 (primeira versão do Android com suporte integral para *tablets*) e superiores. Estes ícones devem estar no formato 32-bit PNGs com canal alfa para transparência, e devem possuir background transparente. As dimensões destes ícones podem ser consultadas de seguida.

Tabela 16 - Dimensões dos *action bar icons* em pixéis

	ldpi (120dpi) *(Low density screen)*	**mdpi** (160dpi) *(Medium density screen)*	**hdpi** (240dpi) *(High density screen)*	**xhdpi** (320dpi) *(Extra-high density screen)*
Tamanho ícone	18x18 px	24x24 px	36x36 px	48x48 px

Os ***status bar icons*** representam notificações da aplicação na barra de estado. Este tipo de ícones sofreram grandes alterações da versão 2.3 para a 3.0. Aqui, apenas estão expostas as características para a versão 2.3. Nestes ícones também é necessário ter em conta a densidade do ecrã. Os posicionamentos e os tamanhos estão apresentados na Tabela 17, em que o 'w' – indica a largura (*width*) e o 'h' – indica a altura (*heigth*).

Tabela 17 - Posicionamento e dimensões dos *status bar icons*

(hdpi) ***screens:*** *Full Asset:* (24w)x(38h) px *Icon:* (24w)x(24h) px	
(mdpi) ***screens:*** *Full Asset:* (16w)x(25h) px *Icon:* (16w)x(16h) px	
(ldpi) ***screens:*** *Full Asset:* (12w)x(19h) px *Icon:* (12w)x(12h) px	

Na Figura 56, é possível observar ícones correctos e incorrectos.

Figura 56 - *Status Bar Icons* correctos e incorrectos

Os ***tab icons*** são elementos que representam uma aba numa interface multi-abas. Podem estar em dois estados: seleccionado ou não seleccionado. Estes devem ter formas simples. As dimensões e posicionamento encontram-se na tabela:

Tabela 18 - Posicionamento e dimensões dos *tab icons*

(hdpi) *screens:* *Full Asset:* 48x48 px *Icon:* 42x42 px	
(mdpi) *screens:* *Full Asset:* 32x32 px *Icon:* 28x28 px	
(ldpi) *screens:* *Full Asset:* 24x24 px *Icon:* 22x22 px	

Alguns exemplos correctos e incorrectos dos ***tab icons*** podem ser observados na Figura 57.

Figura 57 - *Tab Icons* correctos e incorrectos

Os ***dialogue icons*** aparecem nas caixas de diálogo que requerem interacção com o utilizador. Utilizam gradiente de luz e sombras para se fazer destacar em fundos escuros. As dimensões encontram-se na tabela seguinte.

Tabela 19 - Dimensões dos *dialogue icons*

	ldpi (*Low density screen*)	**mdpi** (*Medium density screen*)	**hdpi** (*High density screen*)
Tamanho ícone	24x24 px	32x32 px	48x48 px

Os ***list view icons*** são apresentados nas *Listviews* (listagem de items). São similares aos anteriores, a diferença está no efeito de sombra.

b) Widget design guidelines

Os *widgets* são funcionalidades introduzidas na versão Android 1.5, e bastante melhoradas a partir da versão 3.0. Servem para mostrar informações relevantes das aplicações, em tempo real, no *home screen* do sistema. Estes são constituídos por vários componentes, como se pode verificar na Figura 58.

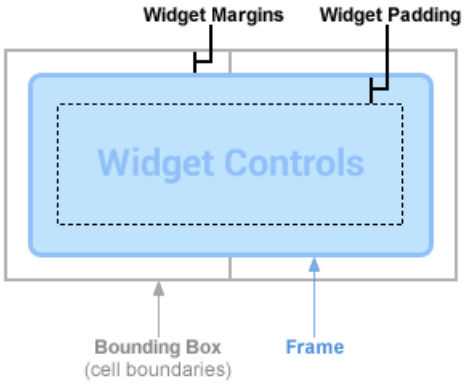


Figura 58 - Componentes dos *widgets*

Nos *widgets*, os atributos como a largura mínima e altura mínima têm bastante importância pois definem a partir do qual o *widget* se torna ilegível. Na Figura 59 pode ser observado um bom exemplo do cálculo de largura e altura mínima.

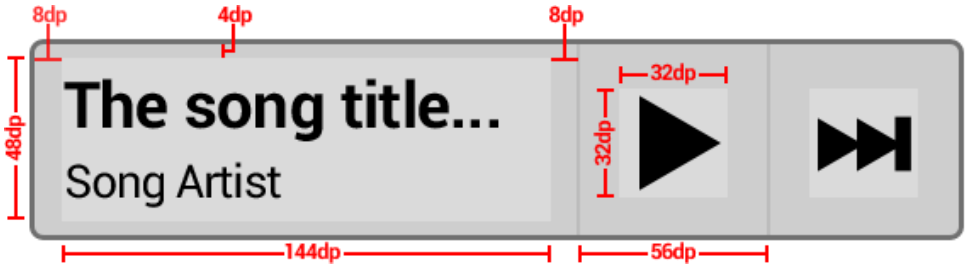


Figura 59 - Exemplo de um *widget* bem dimensionado

Da figura é possível depreender que a dimensões mínimas são dadas por:

minWidth = 144dp + (2 × 8dp) + (2 × 56dp) = 272dp

minHeight = 48dp + (2 × 4dp) = 56dp

c) Activity e Task design guidelines

As *activities* e as *tasks* são dois dos conceitos mais importantes em Android. Muito resumidamente, as *activities* são componentes de uma aplicação que fornecem *screens*

através dos quais os utilizadores podem interagir com o sistema, com o objectivo de realizar as mais diversas tarefa, como tirar uma fotografia, ver um mapa ou enviar um *email*. Estas ficam organizadas numa *stack*[39] quando são criadas, o que proporciona um histórico de navegação linear para as *activities* visitadas pelo utilizador, no qual a tecla BACK tem bastante importância, pois permite voltar para a *activity* anterior que se encontra na *stack*. As *tasks* são uma sequência de *activities* que o utilizador leva a cabo para cumprir um determinado objectivo, como por exemplo, enviar uma mensagem com um anexo. As interrupções das *tasks* podem ter essencialmente duas origens: o utilizador recebe uma notificação e quer actuar sobre ela ou o utilizador decide começar uma nova *task* e eventualmente retornar depois do ponto onde se encontrava.

Devido a esta organização particular, existe algumas regras que devem ser seguidas, principalmente no que toca à navegação entre *activities* e à sua utilização/reutilização. De seguida são apresentados alguns dos aspectos mais importantes relativos a estes conceitos.

- **Iniciar uma *activity* a partir do *Home screen***

A maioria das aplicações são iniciadas no *Home screen*. Quando o utilizador toca no ícone de iniciação da aplicação, a *activity* principal dessa aplicação é iniciada em primeiro plano. Um exemplo pode ser visto na Figura 60.

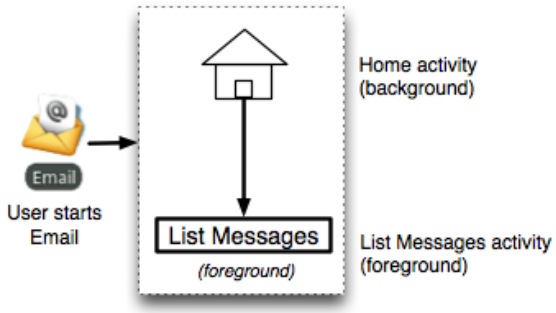


Figura 60 - Diagrama de iniciação de uma *activity* a partir do *Home screen*

- **Sair da *activity* utilizando as teclas BACK e HOME:**

Por defeito, ao pressionar a tecla BACK, a *activity* actual é finalizada e é mostrada a *activity* anterior. A Figura 61 exemplifica este comportamento.

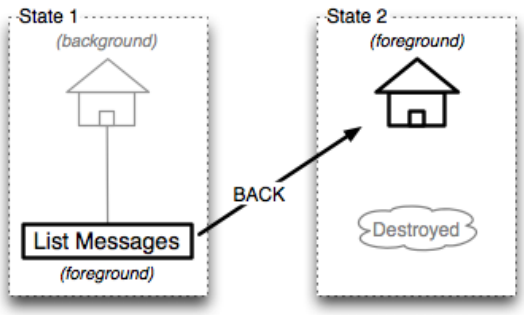


Figura 61 - Efeito tecla BACK

Pelo contrário, ser for pressionada a tecla HOME, a *activity* é apenas parada, colocada em segundo plano na *stack*, e é mostrada na mesma a *activity* anterior, como se pode observar através da Figura 62.

[39] *Activity Stack.*

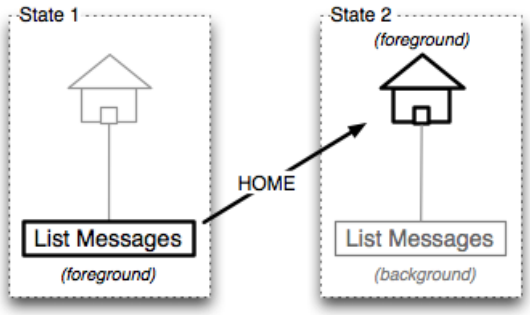


Figura 62 – Efeito tecla HOME

- **Reutilizar uma *activity*:**

Quando uma *activity* A inicia uma *activity* B de outra aplicação, diz-se que a *activity* B é reutilizada, isto porque a A não tem essa funcionalidade mas consegue obter o que pretende através da B. Um exemplo é quando o utilizador pretende adicionar uma imagem num dos contactos e essa imagem encontra-se na galeria de imagens, a *activity* da edição do contacto tem de iniciar a *activity* da galeria de imagens, da forma que se pode observar na Figura 63.

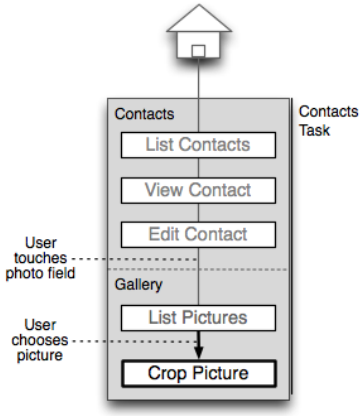


Figura 63 – Reutilização de uma *activity*

- **Multitasking:**

Como já foi referido, podem ser executadas várias tarefas em simultâneo. O utilizador, depois de iniciar uma *activity* pode iniciar outra sem destruir a primeira. Exemplo:

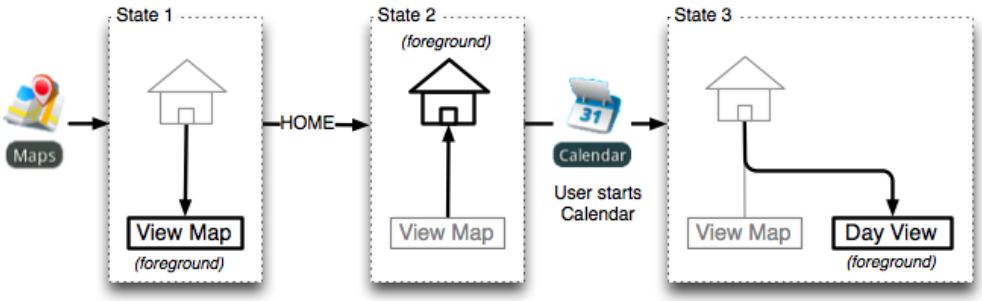


Figura 64 – Funcionamento em *multitasking*

O utilizador inicia uma Map *activity,* mas o carregamento das imagens está lento. Pressiona então na tecla HOME passando para o *Home screen* e enquanto o mapa continua a ser gerado em segundo plano, o utilizador selecciona o calendário, passando este para primeiro plano.

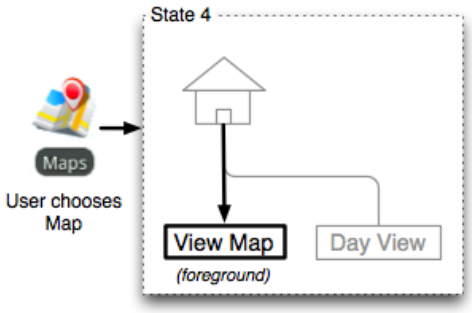


Figura 65 - Funcionamento em *multitasking*

Depois, o utilizador pode voltar a pressionar a tecla HOME (ou BACK), pressionar de novo e Maps e por esta altura já o mapa deve estar completamente carregado e pronto a ser visualizado.

d) Menu design guidelines

Outro tipo de *guidelines* desenvolvidos pelo Android é o que se aplica aos vários menus que são apresentados nas aplicações. Os menus são listas de comandos que podem ser accionados pelos utilizadores, encontram-se geralmente escondidos, e estão acessíveis por botões, teclas ou gestos. Servem para realizar diversas operações, para além de permitir a navegação para outras partes da aplicação ou mesmo para outras aplicações.

Em sistemas Android existem dois tipos de menus que devem ser organizados de acordo com as funcionalidades e com a navegação na aplicação:

- ***Options menu*:** Onde são apresentadas os comandos globais relativos à *activity* actual, ou a partir do qual se pode passar para outra *activity*. Na maioria dos dispositivos, estes menus podem acedidos pressionando o botão MENU. De modo a alargar o número de itens no menu, estes estão subdivididos em dois tipos, os *options icon menu* e os *options expanded menu*. Este tipo de menu pode ser observado no exemplo da Figura 66.

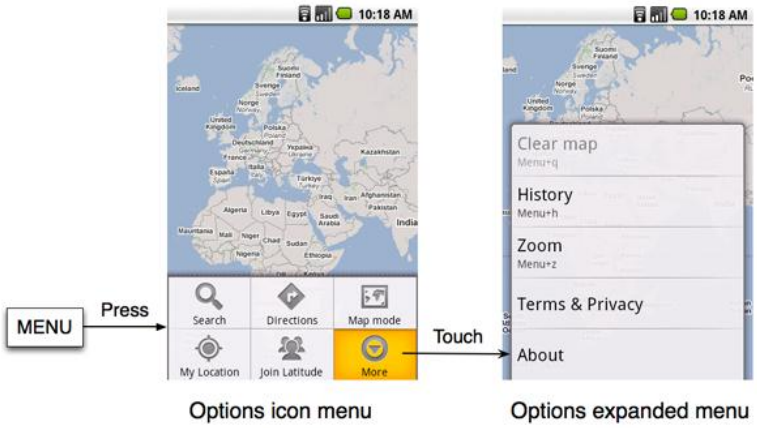


Figura 66 - Exemplo de *Option Menu*

- ***Context menu*:** Trata-se de uma lista de comandos que operam conforme o item seleccionado. É similar ao clique no botão direito do rato. O acesso a estes menus é realizado quando o utilizador faz *touch&hold* sobre um item (se existir). Um exemplo deste tipo de menu encontra-se na Figura 67:

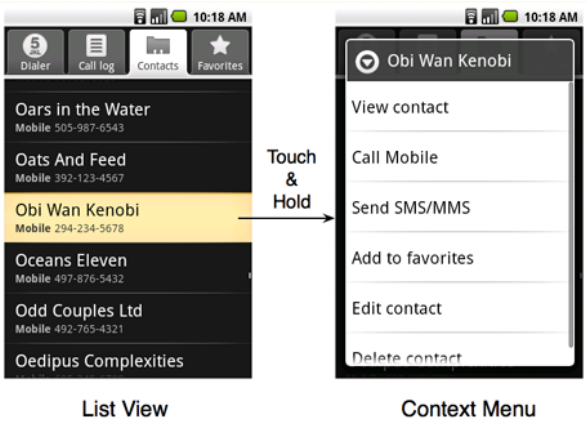


Figura 67 - Exemplo de *Context Menu*

As principais diferenças entre estes dois tipos de menus estão patentes na Figura 68:

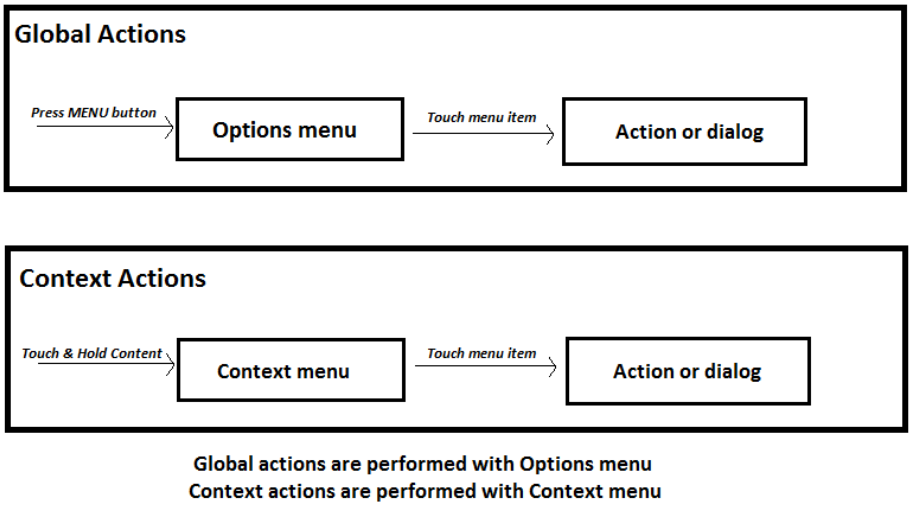


Figura 68 – Principais diferenças entre menus de contexto e menu de opções

3. Suporte para diferentes tipos de ecrã

Como foi referido, os elementos visualizados (botões, ícones, textos, etc.) dependem do *display* em que vão ser visualizados. As características a ter em conta são: tamanho, densidade, orientação, resolução. Na Figura 69, pode-se observar como são categorizados (de forma algo grosseira) os diferentes ecrãs em termos de tamanho (diagonal) e de densidade.

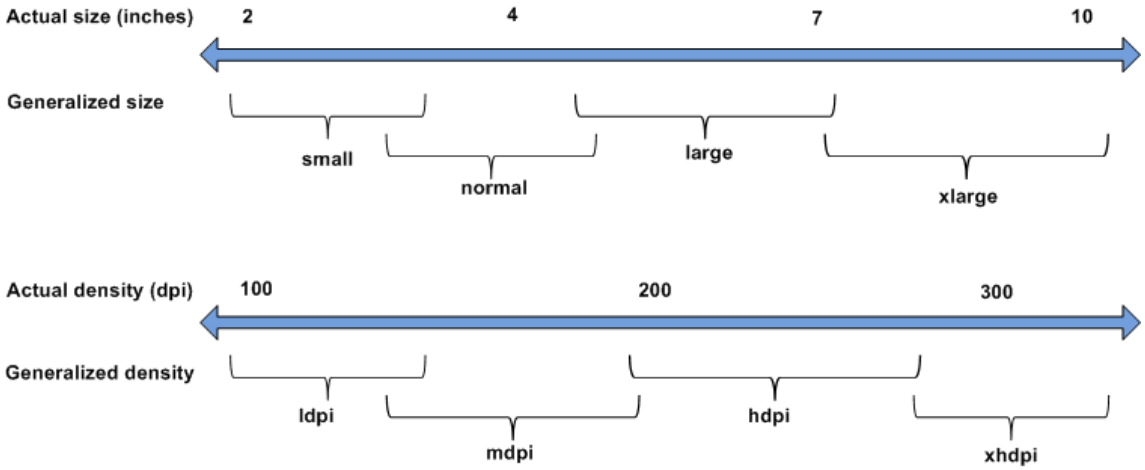


Figura 69 - Classificação dos *displays*

Para se ter uma melhor percepção da influência destas características na visualização dos *layouts*, a Figura 70 e a Figura 71 mostram as diferenças entre uma aplicação que não é independente da densidade do *display* e outra que suporta variações de densidade do ecrã.

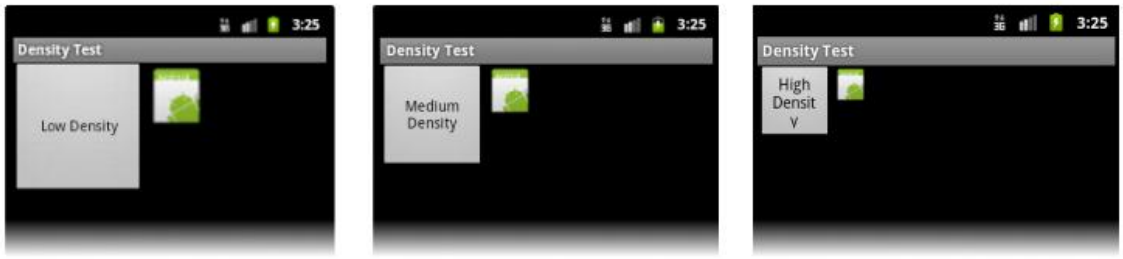


Figura 70 – Exemplo de aplicação que depende da densidade do ecrã

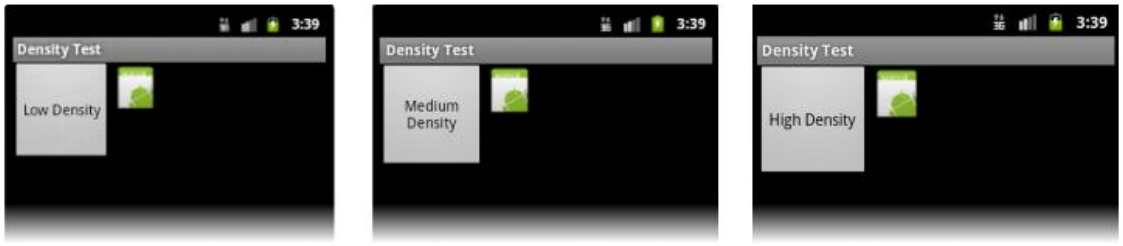


Figura 71 – Exemplo com independência da densidade

Como se pode observar, para diferentes densidades de ecrã (baixa, média e alta), o exemplo que depende da densidade apresenta diferentes resultados, o que pode levar a situações indesejáveis nos *layouts* e na usabilidade da aplicação. Ou seja, no desenvolvimento de uma aplicação, devem ser utilizadas técnicas para que esta se adapte a diferentes tipos de *display*, visto não se saber previamente quais os tipos de ecrã em que a aplicação irá ser executada. Existem essencialmente três técnicas que devem ser utilizadas para a aplicação se poder adaptar a diferentes ecrãs:

- Declarar explicitamente no manifesto da aplicação[40] quais as dimensões de ecrã suportados pela aplicação;
- Fornecer diferentes *layouts* para diferentes dimensões de ecrã;
- Fornecer diferentes imagens *bitmap* para diferentes densidades de *display*.

[40] AndroidManifest.xml – Ficheiro que contêm informação essencial sobre a aplicação.

Como é descrito no primeiro ponto deste anexo, o sistema operativo escolhe em *runtime* o recurso mais adequado para o tipo de ecrã em que está a ser executada a aplicação. Os *layouts* são definidos em XML, e as imagens *bitmap* devem ter as seguintes dimensões para cada densidade:

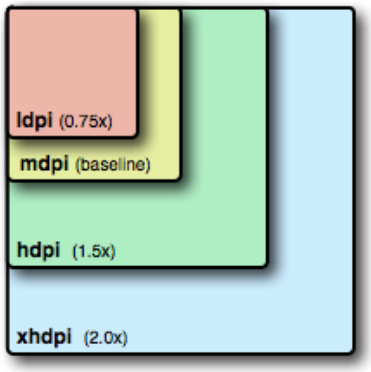


Figura 72 – Tamanhos relativos para imagens bitmaps que suportam as várias densidades

A título de exemplo, se for pretendido utilizar uma imagem de 48x48 pixéis para a densidade média (mdpi), isto significa que é necessário fornecer aos recursos as seguintes imagens:

- 36x36 para ldpi (48*0.75 x 48*0.75)
- 48x48 para mdpi
- 72x72 para hdpi (48*1.5 x 48*1.5)
- 96x96 para xhdpi (48*2.0 x 48*2.0)

4. *Activities* e ciclo de vida de uma aplicação

No contexto do sistema operativo Android, uma *activity* é uma componente da aplicação que permite exibir os diversos *layouts* que compõem o seu *user interface* (e definir o comportamento do elementos que o constituem), possibilitando que os utilizadores possam interagir com a aplicação no sentido de efectuar alguma tarefa, como por exemplo, enviar um *email*, ver um mapa ou tirar uma fotografia. Tipicamente, os *layouts* preenchem o ecrã na sua totalidade. Contudo, também podem ser menores que o ecrã e surgirem com algum efeito de flutuação sobre outro *layout*.

Uma aplicação consiste em múltiplas *activities* interligadas. Tipicamente, uma *activity* é especificada como sendo a "*main*", sendo esta a *activity* apresentada ao utilizador no início da execução da aplicação. Cada *activity* pode iniciar uma nova *activity* de forma a ser executada uma tarefa diferente. Cada vez que uma *activity* é iniciada, esta é adicionada a uma *stack* (designada por *back stack*), onde se encontram também as *activities* previamente inicializadas que ainda não tenham sido eliminadas. Quando uma nova *activity* é iniciada, esta é colocada no top da *back stack*, ficando com o foco da atenção do utilizador (fica em primeiro plano). Sendo a *back stack* utilizada sob o mecanismo de gestão "*last in, first out*", quando o utilizador finaliza a tarefa actual e pressiona o *back button*, a *activity* é removida da *stack* (e é eliminada) e a *activity* anterior é retomada, com o estado anterior do UI a ser restaurado. Este comportamento está representado na Figura 73.

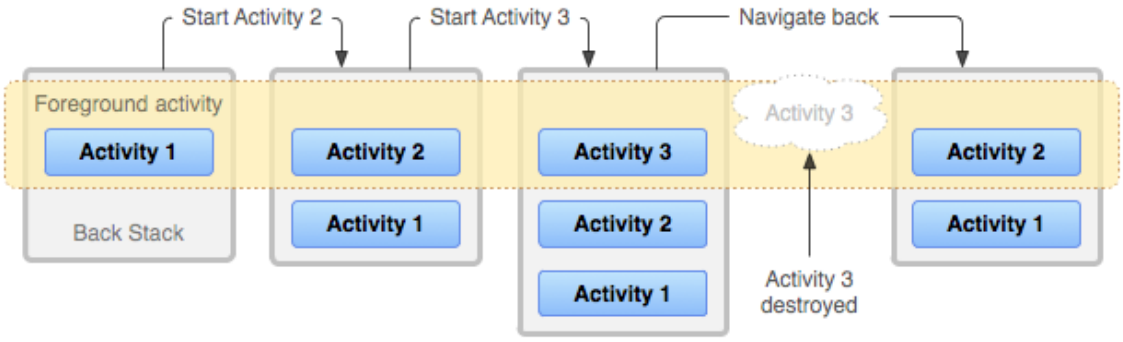


Figura 73 – Manipulação das *activities* ao longo do ciclo de vida de uma aplicação

Fonte: Android Developers [13]

As *activities* podem estar em três estados distintos:

- *Resumed/Running*: a *activity* está em primeiro plano;
- *Paused*: outra *activity* está em *foreground*, no entanto esta continua visível (pode não estar totalmente visível). Em situações de falta de memória extrema em que é necessário libertar memória, o sistema operativo pode eliminar esta *activity*[41];
- *Stopped*: a *activity* já não se encontra visível, ficando em *background,* e pode ser eliminada sempre que seja necessária memória.

[41] Existem mecanismos para salvaguardar e restaurar dados para que não seja perdida informação.

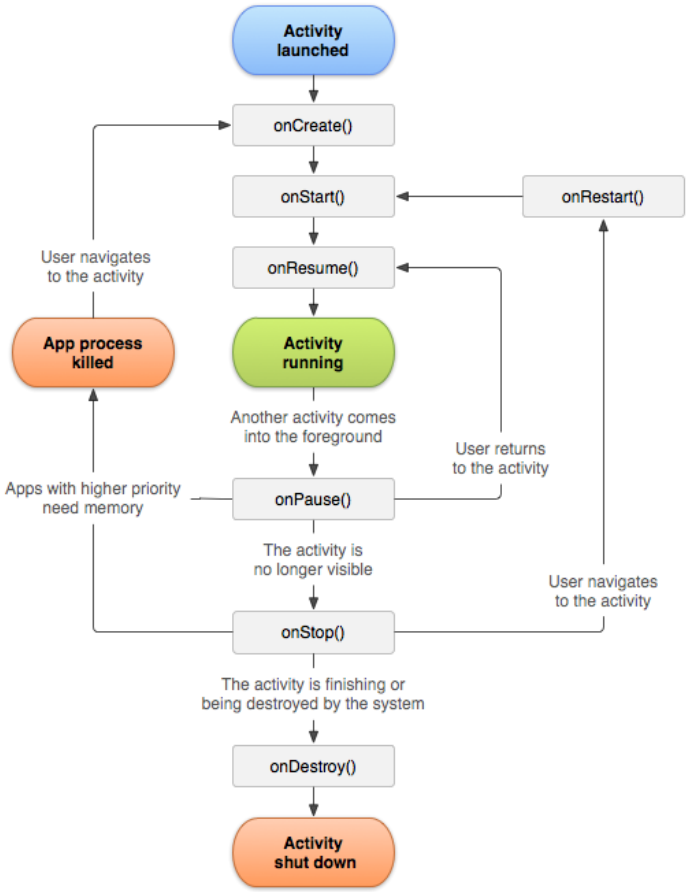


Figura 74 – Ciclo de vida de uma *activity*

Fonte: *Android Developers* [13]

A Figura 74 ilustra o ciclo de vida de uma *activity*. As transições entre estados das *activities* é efectuada com recurso a *callback methods* (por exemplo, *onCreate(), onResume(), onStop()*), que devem ser substituídos (o*verride*) para realizar determinadas acções quando o estado da *activity* é alterado, por exemplo, salvaguardar/restaurar dados, inicializar/libertar recursos.

5. *Threads*

Em Android, cada aplicação é executada num processo separado e tipicamente, todas as componentes da aplicação[42] são executadas no respectivo processo. Contudo, em casos específicos, é possível criar outros processos para manipulação das componentes. Cada processo é composto por pelo menos uma *thread* denominada de "main" *thread* ou UI *thread*. Esta é responsável por manipular os eventos resultantes da interacção com o UI. A utilização de apenas uma *thread* (a UI *thread*) para efectuar todas as operações não é de todo recomendável, pois em operações longas, por exemplo, acesso à rede ou a bases de dados, o UI fica bloqueado deteriorando o UX. Para este tipo de situações, devem ser utilizadas *threads* separadas (*background* ou *worker thread*) para garantir a capacidade de resposta do UI da aplicação. Uma das restrições relativas às *threads* de *background* está relacionada com a impossibilidade manipular elementos do UI no sentido de aplicar o conceito de *thread*

[42]Existem 4 tipos de componentes numa aplicação Android: *Activities, Services, Content Providers, Broadcast Receivers.*

safety[43]. De modo a contornar este problema, a plataforma Android fornece várias soluções das quais se destacam:

- *AsyncTask*: classe que permite realizar operações de forma assíncrona, numa *worker thread*, com a possibilidade de apresentar os resultados na UI *thread*.

- *Handler*: classe que permite enviar mensagens (ou acções a realizar) de uma *thread* para a fila de mensagens da *thread* que instanciou esse *Handler*. Posteriormente, a mensagem é processada pela thread recorrendo a um "*Looper*" e o UI é actualizado. Este comportamento está representado na Figura 75.

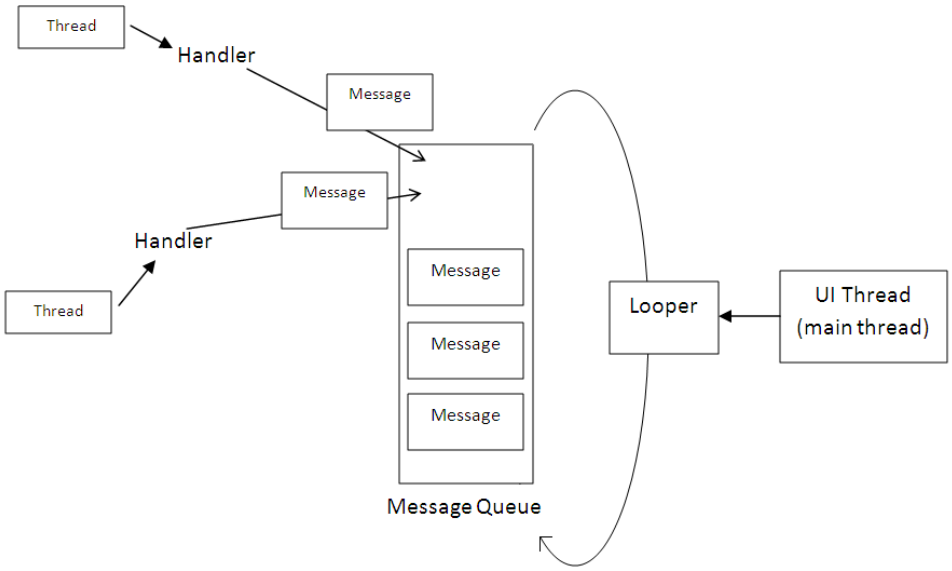


Figura 75 – Utilização de *Handlers* para comunicação entre *threads*

Fonte: *Android – Multithreading in a UI environment* [67]

6. *Intents*

O *intent* é uma estrutura de dados que contem uma descrição abstracta de uma acção a realizar ou também pode conter a descrição de algum evento que tenha ocorrido. Os *intents* podem ser caracterizados como sendo mensagens com capacidade para activar as componentes da aplicação (por exemplo, *activities, services, broadcast receivers*), no contexto da mesma aplicação ou entre aplicações diferentes. Estes permitem também enviar dados entre as componentes.

Os *intents* podem ser utilizados em várias situações específicas, como por exemplo:

- Iniciar uma nova *activity*, com possibilidade de transferir de dados entre a anterior e a seguinte;
- Iniciar uma nova chamada telefónica ou o envio de uma mensagem de texto;
- Iniciar o *browser* para acesso à Internet;
- Notificar a recepção de uma nova mensagem de texto;

[43]Manipulação segura de dados partilhados em ambientes com múltiplas *threads* em execução.

7. Persistência de dados

A plataforma Android oferece várias soluções para persistência dos dados das aplicações. A escolha por uma das soluções depende das necessidades específicas da aplicação. Existem algumas situações a analisar para a escolha por uma das soluções, tais como se os dados devem ser privados ou acessíveis a outras aplicações (e ao utilizador) e a quantidade de espaço necessário. De seguida, são apresentados brevemente as soluções possíveis.

- *Shared Preferences*: Esta opção permite salvaguardar e restaurar dados do tipo primitivo (por exemplo, *booleans, floats, ints*) no formato *key-value*. É geralmente usado para guardar preferências de utilizador para a aplicação (pouca informação). Por exemplo, pode ser utilizado para guardar dados de autenticação, ou para definir se uma funcionalidade da aplicação está ou não activa.
- Armazenamento interno: Através desta solução, é possível guardar ficheiros na memória interna do dispositivo. Tipicamente, estes ficheiros não podem ser acedidos directamente por outras aplicações ou pelo utilizador e quando a aplicação é desinstalada, os ficheiros são removidos.
- Armazenamento externo: Esta opção possibilita guardar grandes quantidades de informação em zonas de memória do dispositivo que são partilhadas, por exemplo, num cartão SD. Ou seja, os dados guardados nestas zonas são acessíveis por outras aplicações e pelo utilizador, pelo que os dados podem ser alterados sem aviso prévio.
- Base de dados SQLite: A plataforma Android fornece suporte para bases de dados SQLite e qualquer base de dados criada é acessível pelo nome, mas apenas no contexto da aplicação.
- Conexão à rede: No caso de existir ligação à rede, é possível salvaguardar e recuperar dados através de serviços *web-based*.

Anexo 5: Procedimento de formulação de ícones

1. Ícones representativos do estado dos dispositivos presentes na rede

A abordagem seguida para este tipo de ícones consiste na junção dos efeitos visuais usados nos *Menu Icons* (sombras, gradientes, cores, etc.), com as dimensões sugeridas para os *List View Icons*. Os efeitos e as dimensões utilizados podem ser observados na Tabela 20 e na Tabela 21.

Tabela 20 - Efeitos aplicados nos ícones

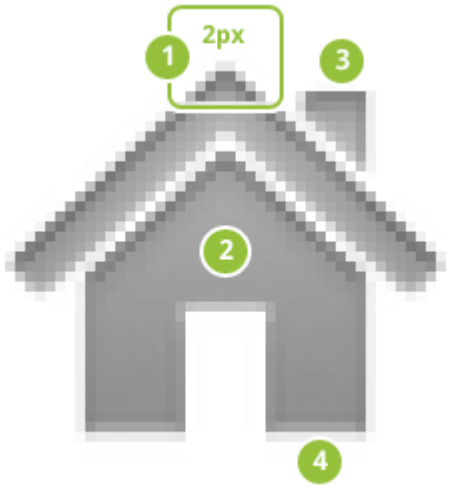
	***1. Corner rounding:***	2 pixel corner radius, when appropriate
	2. Fill gradient:	90°, from #8C8C8C to #B2B2B2
	3. Inner shadow:	#000000, 20% opacity angle 90° distance 2px size 2px
	4. Inner bevel:	depth 1% direction down size 0px angle 90° altitude 10° highlight #FFFFFF, 70% opacity shadow #000000, 25% opacity

Tabela 21 – Dimensões dos ícones

Low density screen *(ldpi)*	**Medium density screen *(mdpi)***	**High density screen *(hdpi)***
24 x 24 px	32 x 32 px	48 x 48 px

Para além dos efeitos apresentados, optou-se por desenhar os ícones com cores adicionais de forma a representar melhor o estado dos diversos dispositivos. Foram escolhidas como cores adicionais o verde (#009444[44]), o amarelo (#FFF200) e o vermelho (#ED1C24).

Os *softwares* utilizados para a edição dos ícones foram: o "Adobe Illustrator"[45], o "Adobe Fireworks"[46] e o "Adobe Photoshop"[47]. Cada um deles tem uma função específica da formulação dos ícones. Tal como é recomendado nos *guidelines*, utilizou-se o Illustrator para

[44] Valor hexadecimal representativo das componentes R-*red*, G-*green*, B-*blue*.
[45] Versão *trial*: http://www.adobe.com/cfusion/tdrc/index.cfm?product=illustrator.
[46] Versão *trial*: http://www.adobe.com/cfusion/tdrc/index.cfm?product=fireworks.
[47] Versão *trial*: http://www.adobe.com/cfusion/tdrc/index.cfm?product=photoshop.

criar as formas básicas dos ícones, sendo estas importadas para o Photoshop, de modo a aplicar os efeitos e estilos pré-definidos. O Fireworks foi utilizado para a exportação das formas dos ícones entre o Illustrator e o Photoshop de modo a manter a máxima capacidade de edição da imagem. Este processo consiste em guardar as formas no Illustrator no formato por defeito do Adobe Illustrator (*.ai), abrir este ficheiro com o Fireworks e aí guardar com o formato do Photoshop (*.psd), sendo agora possível editar convenientemente a imagem no Photoshop. Se se utilizar a importação directa[48] entre o Illustrator e o Photoshop, a imagem ficaria rasterizada[49], em vez de continuar vectorial (como recomendado), o que diminui a sua capacidade de edição.

De seguida é apresentado um exemplo concreto de formulação de um ícone, passo a passo, até chegar ao resultado final. O ícone do exemplo é o da tomada eléctrica.

1. Painel inicial do Adobe Illustrator

A Figura 76 apresenta o painel inicial do Illustrator.

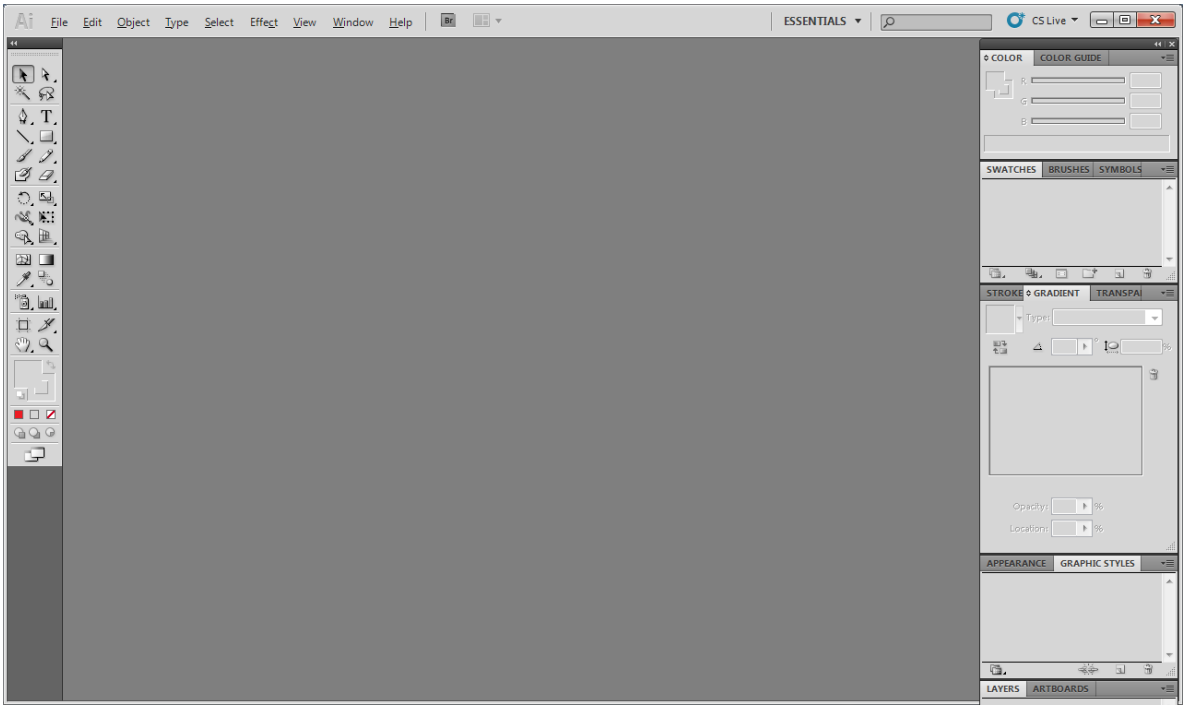


Figura 76 - Painel inicial do Adobe Illustrator CS5

2. Criação de novo documento

O primeiro passo a realizar consiste na criação de um novo documento com dimensões com 250 x 250 px, por exemplo. Estas dimensões são maiores que as anteriormente mencionadas pois é o recomendado, sendo depois estas formas fáceis de redimensionar para as dimensões correctas. A Figura 77 ilustra isso mesmo.

[48] No Illustrator guardar com a extensão *.psd.
[49] Pixéis ou pontos.

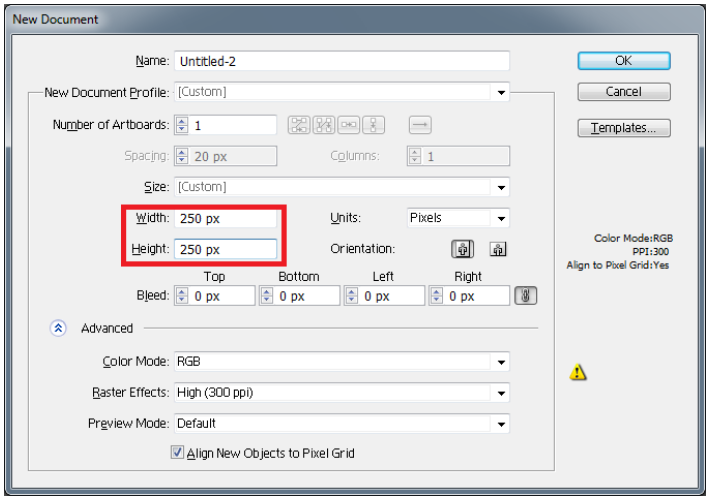


Figura 77 – Criação de um novo documento no Illustrator

3. Formas e cores básicas

Utilizando as diversas ferramentas presentes no *software*, que podem ser encontradas do lado esquerdo e direito, procede-se à criação das formas básicas, preenchendo os espaços com as cores desejadas (cinzento e branco/verde/amarelo/vermelho). O cinzento será alterado quando forem aplicados os efeitos já referidos.

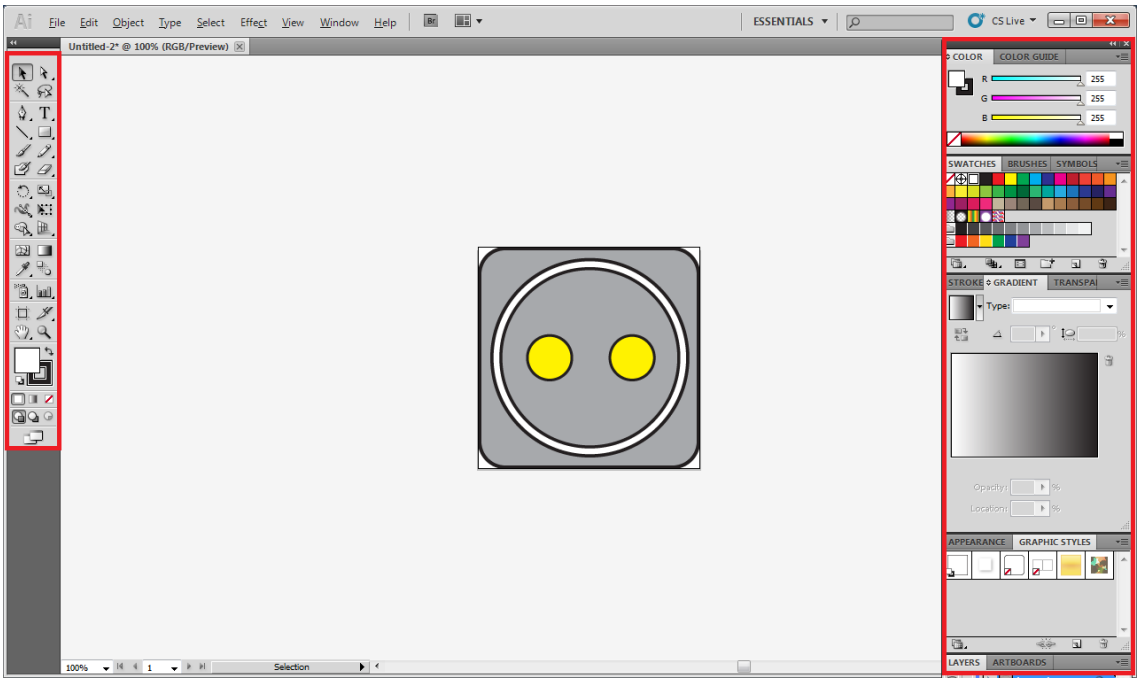


Figura 78 – Desenho das formas básicas no ícone

Depois de criar a forma básica, guarda-se no formato *.ai.

4. Importação para Photoshop

Como já foi referido, utiliza-se o Fireworks para importar para o Photoshop.

Com o Fireworks abre-se o ficheiro previamente guardado e volta-se a guardar com a extensão *.psd.

5. Aplicação dos efeitos no Photoshop

De seguida, abre-se o ficheiro anterior utilizando o Photoshop.

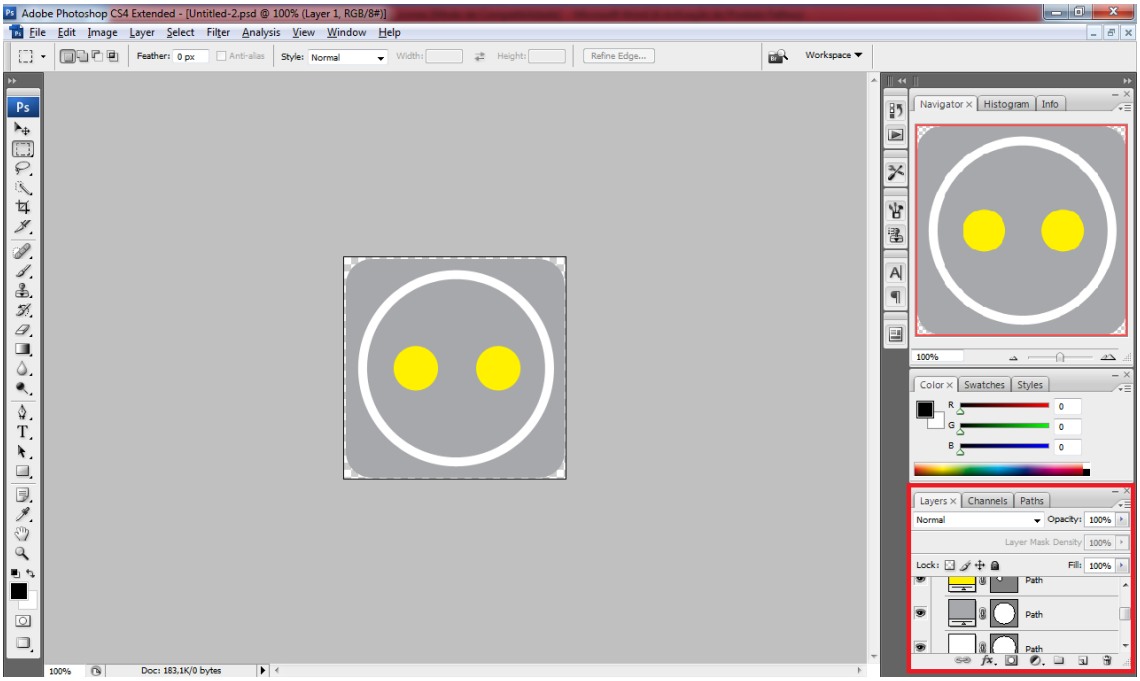


Figura 79 – Painel do Photoshop

Como se pode observar, no canto inferior direito encontram-se os vários componentes das diferentes camadas (*layers*) da imagem. Seleccionando os componentes a cinzento (um de cada vez), clicando no botão direito do rato, escolhe-se a opção "*Blending Options…*". Surge então o painel de aplicação dos diversos efeitos:

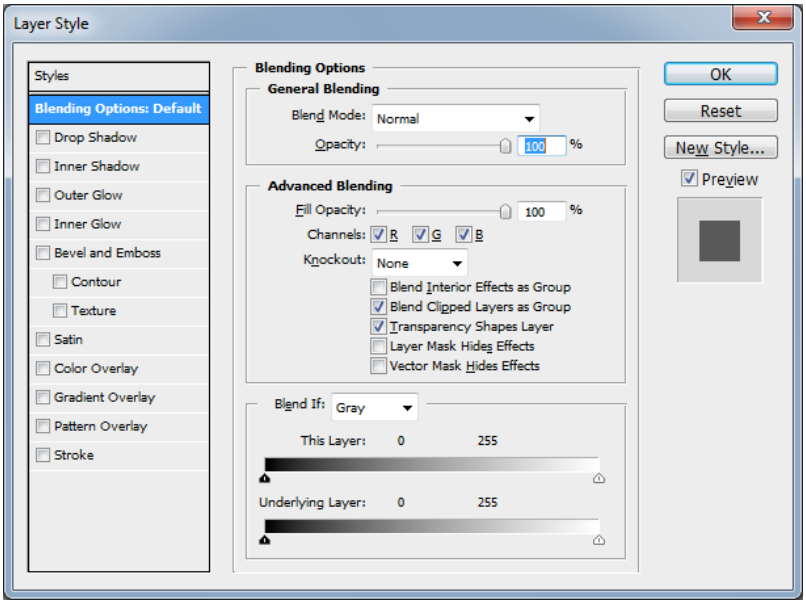


Figura 80 – Aplicação dos efeitos na imagem

É neste ponto do processo que os *templates* fornecidos pela Google são úteis. Abrindo no Photoshop o *template* correspondente aos ícones do tipo Menu, obtém-se os valores dos efeitos aplicados nesses *templates* e deve-se usar os mesmos valores no painel de configuração de efeitos (*Layer Style*) do ícone a criar.

Depois de aplicar os efeitos para todos os componentes a cinzento, é necessário unir (*Merge*) as diferentes camadas. Para isso pode-se usar o atalho de teclado "CTRL+SHIFT+E". Por fim, pode-se optar por remover o *background* da parte que se encontra a branco utilizando a ferramenta "*Magic Eraser Tool*". O resultado é apresentado na Figura 81.

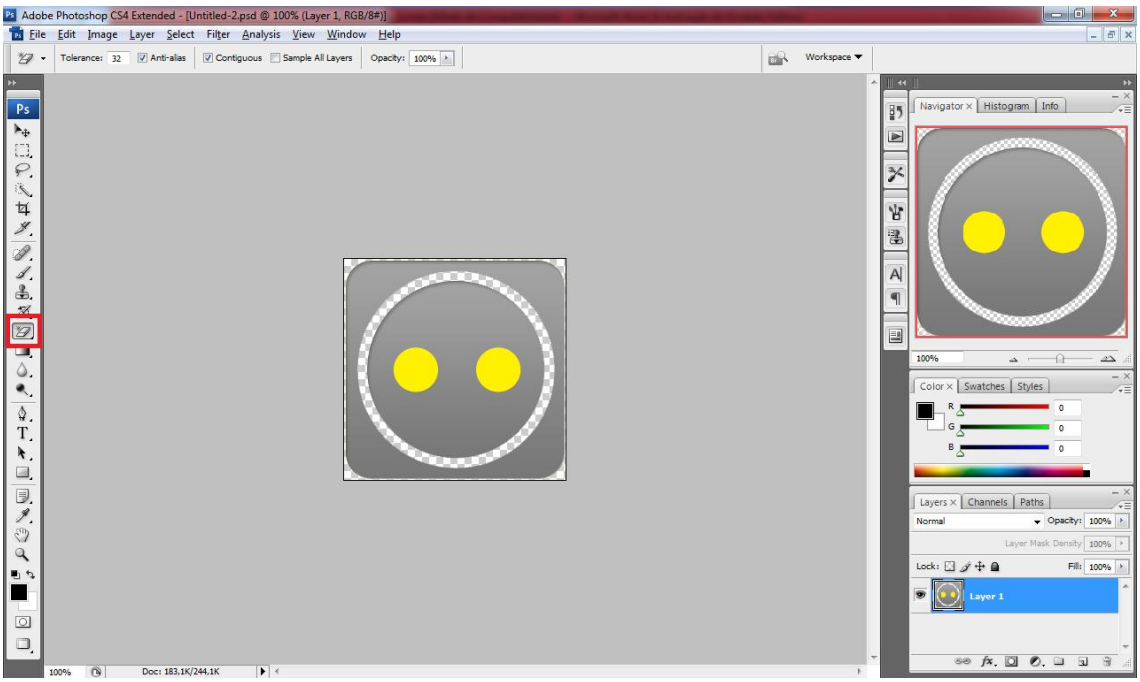


Figura 81 – Aspecto final da imagem

Esta imagem deve ser guardada em formato *.png.

6. **Redimensionamento**

Para finalizar o processo é necessário reduzir as dimensões da imagem para as dimensões anteriormente indicadas, de modo a optimizar a sua visualização em ecrãs com diversas densidades.

Para isso, recorrer-se ao Illustrator, criam-se 3 novos documentos com dimensões 24 x 24px, 32 x 32px e 48 x 48px. Arrasta-se para cada um dos documentos o PNG criado e ajusta-se os limites da imagem de modo a estes coincidirem com os limites do documento. Por último, salvaguarda-se os documentos em formato *.png, ficando assim com o ícone apropriado para cada tipo de densidade.

2. Ícones presentes nos botões de selecção da funcionalidade da aplicação

A abordagem utilizada para este tipo de ícones é em tudo semelhante á anterior. A única diferença está no facto de se ter optado em não usar cores adicionais neste tipo de ícones, ficando apenas em tons de cinzento e branco.

3. Ícones das opções dos menus quando se usa a tecla "MENU"

Relativamente a este tipo de ícones, a abordagem usada consiste em recorrer a uma ferramenta *online* que facilita a criação de diversos tipos de ícones para a plataforma Android. Esta ferramenta é denominada de "Android Asset Studio"[68] e é possível gerar ícones que respeitam os efeitos definidos pela Google a partir de imagens fornecidas, imagens padrão deste tipo de plataforma (*clipart*) e a partir de texto.

Para exemplificar o procedimento, é apresentado de seguida como foi obtido um dos ícones para os menus presentes na aplicação.

Selecciona-se a opção "*Menu icons*" no *site* do "Android Asset Studio" e escolhe-se a imagem desejada em "*Clipart*", que servirá de base para o ícone a ser gerado, como demonstrado na Figura 82:

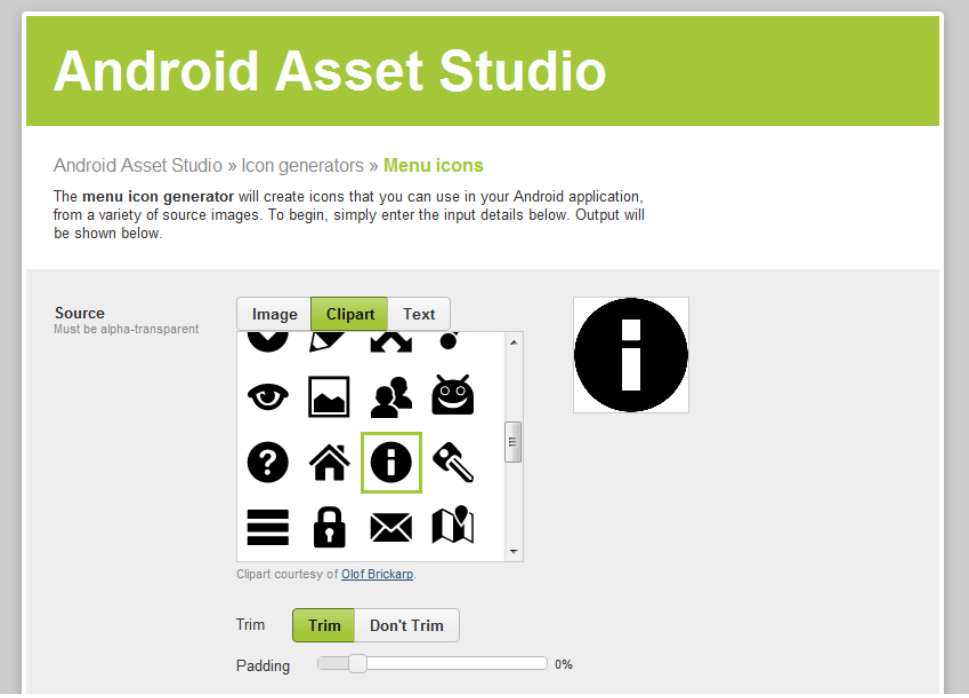


Figura 82 – Android Asset Studio 1

Depois podem ser configuradas algumas opções de ajuste do ícone e no fim da página pode-se observar o aspecto final do ícone para as diferentes densidades de ecrã, como se apresenta na Figura 83:

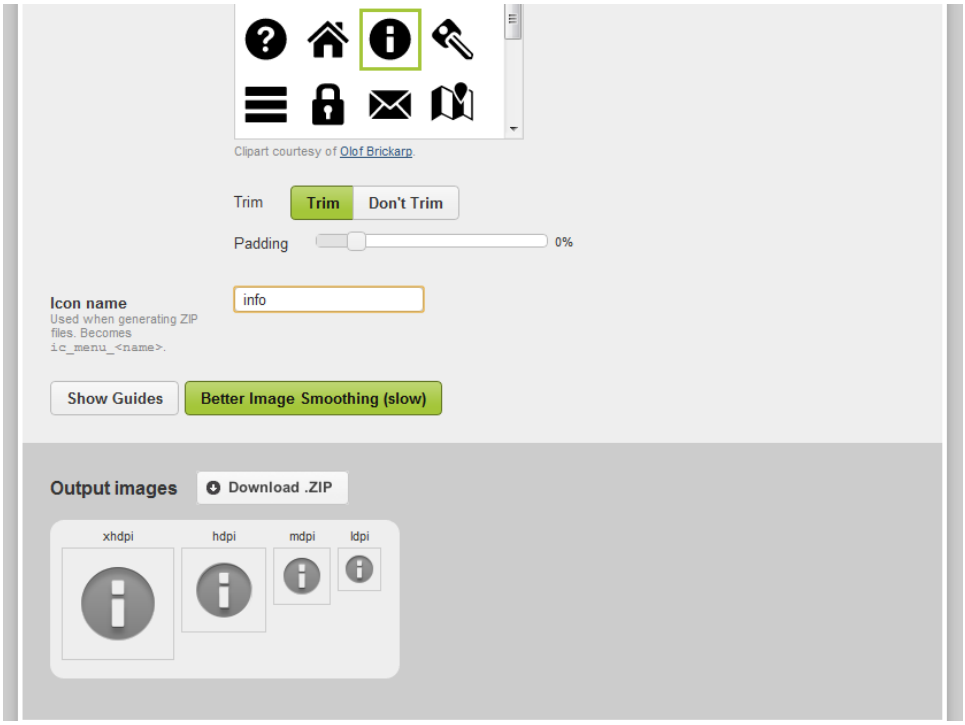


Figura 83 – Android Asset Studio 2

Por fim, basta fazer o download do ficheiro *.zip para se obter o ícone para as diversas densidades.

Anexo 6: *Layouts* para tipos de ecrãs diferentes

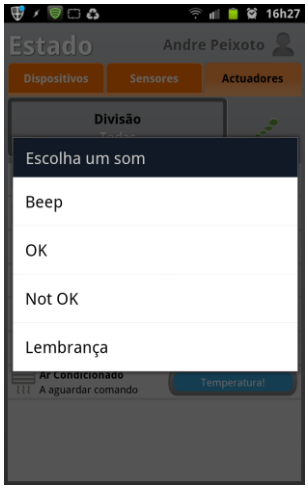
Neste anexo são apresentados os *layouts* das duas aplicações (pré-piloto e piloto) testadas em dois dispositivos com características de ecrãs diferentes:

- Samsung Galaxy Note: *large size* e *high density*;
- Assus Eee Transformer Prime: *extra-large size* e *high density*;

Pré-piloto (Micro I/O)

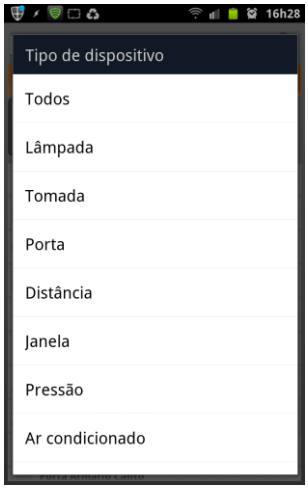
- ***Large size, high density***


Estado
Andre Peixoto
Dispositivos
Sensores
Actuadores
Divisão
Todas
Informação adicional
Última actualização 09:46 30-03-2012
Divisão IDT
Sector Micro I/O
Nível da bateria 0
OK


Estado
Andre Peixoto
Dispositivos
Sensores
Actuadores
Divisão
Escolha um som
Beep
OK
Not OK
Lembrança


Estado
Andre Peixoto
Dispositivos
Sensores
Actuadores
Divisão
Todas
Porta Armario Meio
Desconhecido/inactivo
Tomada Ventoinha
Ligado(a)
Desligar!
Sensor Ventoinha
Distância: 1
Tomada Janela
Desligado(a)
Ligar!
Lampada Rua
Ligado(a)
Desligar!
Lampada Central
Desligado(a)
Ligar!
Sensor Mesa Esq
Início
Actualizar
Visualização
Ordenação
Configurações
Mais


Estado
Andre Peixoto
Dispositivos
Sensores
Actuadores
Divisão
Todas
Visualização
Divisão
Tipo de dispositivo
Sector


Tipo de dispositivo
Todos
Lâmpada
Tomada
Porta
Distância
Janela
Pressão
Ar condicionado


Estado
Andre Peixoto
Dispositivos
Sensores
Actuadores
Tipo de dispositivo
Lâmpada
Lampada Rua
Ligado(a)
Desligar!
Lampada Central
Desligado(a)
Ligar!
Lampada Solis
Ligado(a)
Desligar!

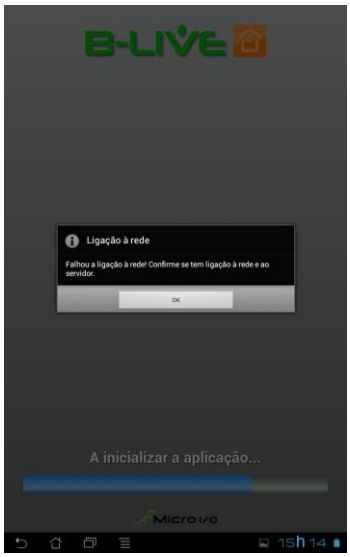

B-LIVE
Utilizador: peixoto
Palavra-chave: •••••••
Relembrar dados
Iniciar Sessão

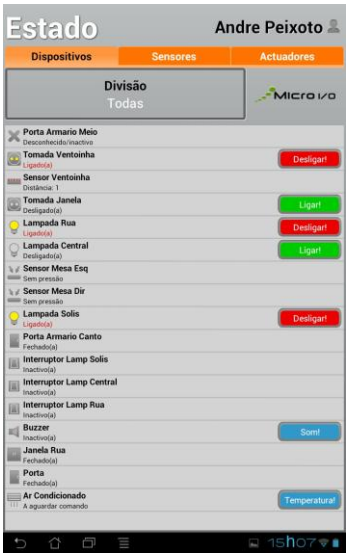

B-LIVE
Andre Peixoto
Estado
Localização
Monitorização
Histórico
Micro I/O

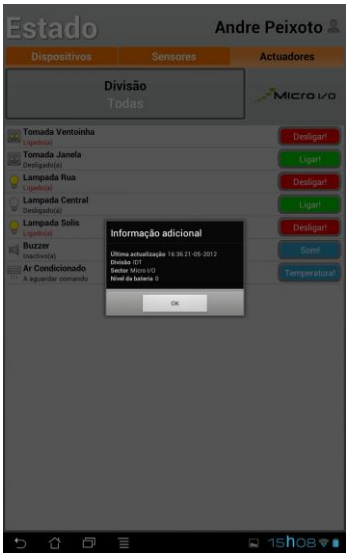

Estado
Andre Peixoto
Divisão
Todas
Micro I/O
B-LIVE
Dispositivos
Sensores
Actuadores
Porta Armario Meio
Desconhecido/inactivo
Tomada Ventoinha
Ligado(a)
Desligar!
Sensor Ventoinha
Distância: 1
Tomada Janela
Desligado(a)
Ligar!
Lampada Rua
Ligado(a)
Desligar!
Lampada Central
Desligado(a)
Ligar!
Sensor Mesa Esq
Sem pressão

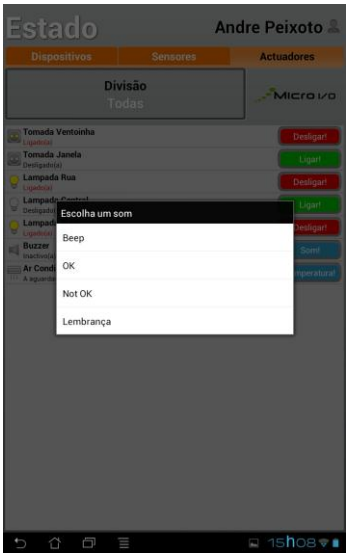

Estado
Dispositivos
Sensores
Actuadores
Tomada Ventoinha
Desligar!
Informação adicional
Última actualização 16:36 21-05-2012
Divisão IDT
Sector Micro I/O
Nível da bateria 0
OK
Ligar!
Desligar!
Ligar!
Desligar!
Som!
Ar Condicionado
A aguardar comando
Temperatura!
B-LIVE

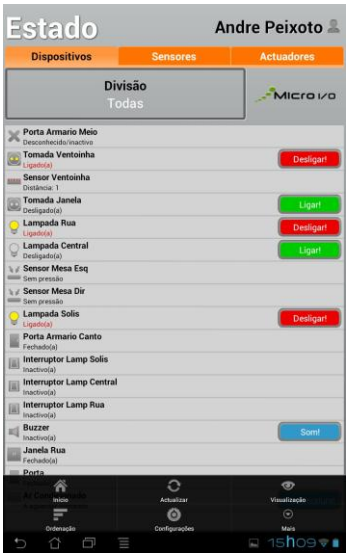
- ***Extra-large size, high density***












Estado
Andre Peixoto
Dispositivos
Sensores
Actuadores
Divisão
Todas
Visualização
Divisão
Tipo de dispositivo
Sector


Estado
Andre Peixoto
Dispositivos
Sensores
Actuadores
Tipo de dispositivo
Todos
Tipo de dispositivo
Todos
Lâmpada
Tomada
Porta
Distância
Janela
Pressão
Ar condicionado
Buzzer
Interruptor


Estado
Andre Peixoto
Dispositivos
Sensores
Actuadores
Tipo de dispositivo
Lâmpada
Lampada Central
Lampada Rua
Lampada Solis
Ligar!
Desligar!


B-LIVE
Utilizador:
peixoto
Palavra-chave:
Relembrar dados
Iniciar Sessão


Estado
Andre Peixoto
Dispositivos
Sensores
Actuadores
Divisão
Todas
B-LIVE


Estado
Andre Peixoto
Escolha uma temperatura
Ligar
Desligar
16º
18º
20º
22º
24º
26º
28º
30º
B-LIVE


Estado
Andre Peixoto
Dispositivos
Sensores
Actuadores
Divisão
Todas
Tomada Ventoinha
Tomada Janela
Lampada Rua
Lampada Central
Lampada Solis
Buzzer
Ar Condicionado
Ligar!
Desligar!
Som!
Temperatura!

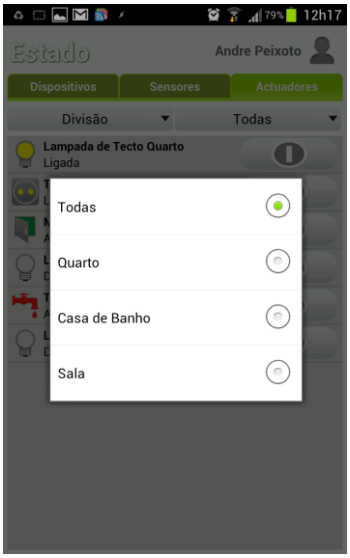
Piloto (ESSUA)

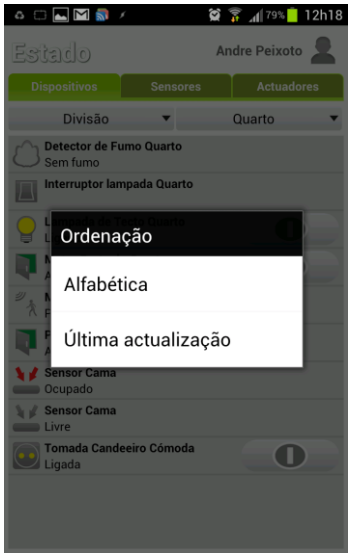
- ***Large size, high density***

Nota: Devido a uma actualização dos *qualifiers* (permite a especificação de *resources* para cada tipo de configuração do dispositivo, por exemplo, tamanho e densidade do ecrã) verificada entre as versões 2.3 e 3.0 da plataforma Android (o dispositivo estava a nesta altura a executar a versão 4.0.3 do sistema operativo), os *layouts* apresentados nesta secção não apresentam a optimização adequada do espaço disponível. Este aspecto será rectificado futuramente.


Estado
Andre Peixoto
Dispositivos
Sensores
Actuadores
Sector
Todos
Lampada de Tecto Quarto
Ligada
Tomada Candeeiro Cómoda
Andre Peixoto
Informações
Terminar Sessão

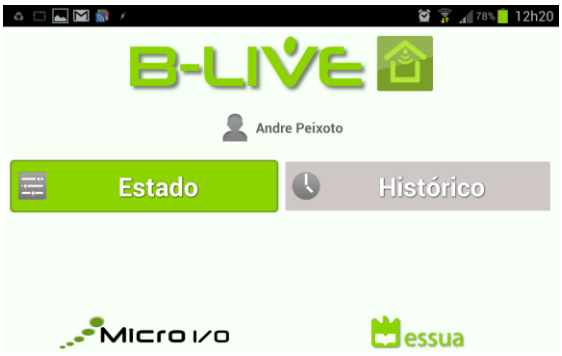

Estado
Andre Peixoto
Dispositivos
Sensores
Actuadores
Sector
Todos
Lampada de Tecto Quarto
Ligada
Andre Peixoto
Data de nascimento 1989-06-01
Género Masculino
Telefone 999993424
Email peixoto@gmail.com
Nacionalidade PT
OK


Estado
Andre Peixoto
Dispositivos
Sensores
Actuadores
Divisão
Todas
Lampada de Tecto Quarto
Ligada
Todas
Quarto
Casa de Banho
Sala


Estado
Andre Peixoto
Dispositivos
Sensores
Actuadores
Divisão
Quarto
Detector de Fumo Quarto
Sem fumo
Interruptor lampada Quarto
Ordenação
Alfabética
Última actualização
Sensor Cama
Ocupado
Sensor Cama
Livre
Tomada Candeeiro Cómoda
Ligada


Estado
Andre Peixoto
Dispositivos
Sensores
Actuadores
Divisão
Quarto
Detector de Fumo Quarto
Sem fumo
Interruptor lampada Quarto
Lampada de Tecto Quarto
Ligada
Motor Porta do Quarto
Aberta
Movimento Quarto
Parado
Porta do Quarto
Aberta
Sensor Cama
Ocupado
Sensor Cama
Livre
Tomada Candeeiro Cómoda
Ligada


B-LIVE
peixoto
Entrar automaticamente
Iniciar Sessão


B-LIVE
Andre Peixoto
Estado
Histórico
Micro I/O
essua

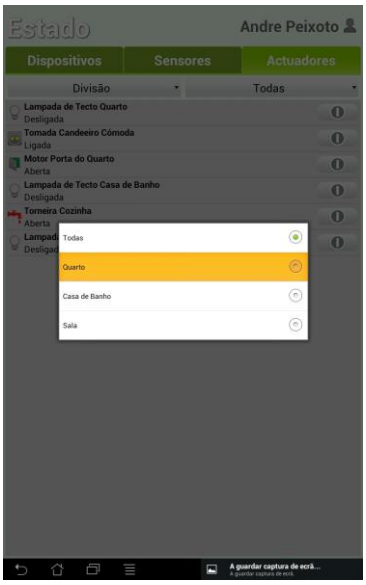
- ***Extra-large size, high density***

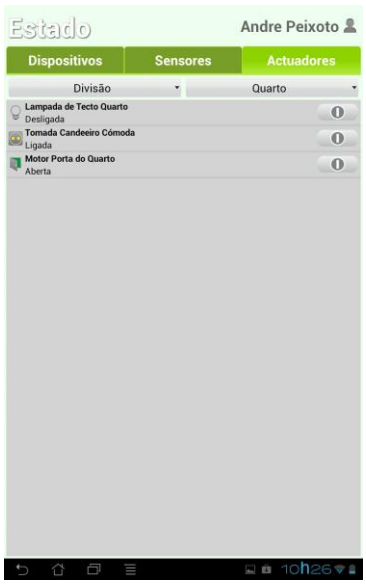

B-LIVE
Andre Peixoto
Andre Peixoto
Informações
Terminar Sessão
Histórico
Micro I/O
essua


B-LIVE
Andre Peixoto
Andre Peixoto
OK
Histórico
Micro I/O
essua


Estado
Andre Peixoto
Dispositivos
Sensores
Actuadores
Sector
Todos
Movimento Quarto
Parado
Interruptor lampada Quarto
Detector de Fumo Quarto
Sem fumo
Lampada de Tecto Quarto
Ligada
Tomada Candeeiro Cómoda
Ligada
Motor Porta do Quarto
Aberta
Porta da Casa de banho
Aberta
Sensor Espelho
0.173 m
Movimento Casa de banho
Parado
Interruptor lampada Casa de Banho
Lampada de Tecto Casa de Banho
Desligada
Sensor Sofa
Livre
Sensor Sofa
Livre
Temperatura Sala
26.2 ºC
Detector de Fumo Sala
Actualizar
Ordenação
Sobre
Ajuda


Estado
Andre Peixoto
Dispositivos
Sensores
Actuadores
Divisão
Todas
Lampada de Tecto Quarto
Desligada
Tomada Candeeiro Cómoda
Ligada
Motor Porta do Quarto
Aberta
Lampada de Tecto Casa de Banho
Desligada
Torneira Cozinha
Aberta
Lampada
Desligada


Estado
Andre Peixoto
Dispositivos
Sensores
Actuadores
Divisão
Todas
Lampada de Tecto Quarto
Desligada
Tomada Candeeiro Cómoda
Ligada
Motor Porta do Quarto
Aberta
Lampada de Tecto Casa de Banho
Desligada
Torneira Cozinha
Aberta
Todas
Quarto
Casa de Banho
Sala


Estado
Andre Peixoto
Dispositivos
Sensores
Actuadores
Divisão
Quarto
Lampada de Tecto Quarto
Desligada
Tomada Candeeiro Cómoda
Ligada
Motor Porta do Quarto
Aberta


B-LIVE
Nome do Utilizador
Palavra-Chave
Entrar automaticamente
Iniciar Sessão
Micro I/O
essua


B-LIVE
Andre Peixoto
Estado
Histórico
Micro I/O
essua


Estado
Dispositivos
Sensores
Actuadores
Andre Peixoto
Sector
Todos
Lampada de Tecto Quarto
Desligada
Tomada Candeeiro Cómoda
Ligada
Motor Porta do Quarto
Aberta
Lampada de Tecto Casa de Banho
Desligada
10h22


Estado
Dispositivos
Sensores
Actuadores
Andre Peixoto
Divisão
Todas
Lampada de Tecto Quarto
Desligada
Tomada Candeeiro Cómoda
Ligada
Motor Porta do Quarto
Aberta
Informações adicionais
OK
10h26

Anexo 7: *Web Services* SOAP

O SOAP (*Simple Object Access Protocol*) é um protocolo baseado em XML (*eXtensible Markup Language*) que possibilita a troca de informação entre aplicações com recurso à tecnologia http (HyperText Transfer Protocol). Pode ser visto também como:

- Um protocolo de comunicação;
- Um formato para a troca de mensagens;
- Uma forma de comunicar através da Internet;
- Independente da plataforma e da linguagem utilizada pelas aplicações;

O protocolo SOAP permite comunicar entre aplicações utilizando *Remote Procedure Calls* (RPC), sendo possível desta forma usar este protocolo para aceder a *Web Services* (WS).

Todas a mensagem SOAP são constituídas pelos seguintes elementos (Figura 84):

- *Envelope*: É o elemento raiz do documento XML. Contém as declarações dos *namespaces* e estilos de codificação da mensagem;
- *Header*: Este elemento é opcional e contém informação adicional, como por exemplo, se a mensagem deve ser processada por nós intermediários na rede;
- *Body*: Contém a informação a ser transportada.

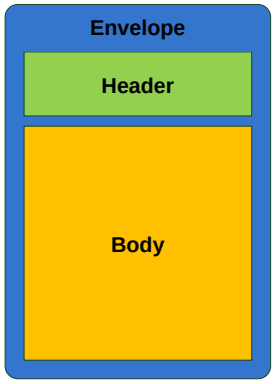


Figura 84 – Envelope de uma mensagem SOAP

Um elemento importante no consumo de WS é o documento WSDL (*Web Service Description Language*). Este documento, baseado em linguagem XML é utilizado para descrever todos os aspectos do WS, por exemplo, o formato dos métodos a serem invocados e dos parâmetros a serem passados. Cada WS tem um ficheiro WSDL associado.

Na Figura 85 é possível verificar um cenário típico de utilização de WS. Uma aplicação cliente localiza o serviço remoto (no servidor) através do documento WSDL disponibilizado e invoca os seus serviços através de RPC. O WS recebe a chamada, efectua o processamento e envia a resposta de volta para a aplicação cliente.

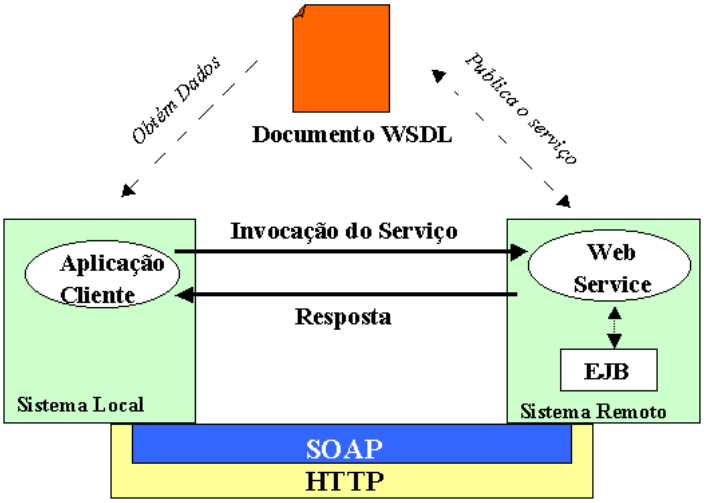


Figura 85 – Cenário de utilização de SOAP para acesso a *Web Services*

Fonte: Web Services, SOAP e Aplicações Web [69]

De modo a exemplificar o processo de consumo de um *Web Service* SOAP em Android, é apresentado de seguida um exemplo prático cujo objectivo é invocar um WS disponibilizado na Internet que converte um valor de peso de uma unidade para outra (baseado no exemplo de [70]). O WSDL deste serviço encontra-se disponível no seguinte *link*:

http://www.webservicex.net/ConvertWeight.asmx?WSDL

A plataforma Android não suporta o protocolo SOAP nativamente, pelo que é necessário recorrer a APIs desenvolvidas por terceiros. Uma das mais populares é a "kSOAP2". Depois de integrar devidamente esta API no projecto e adicionar as permissões de acesso à Internet no ficheiro AndroidManifest.xml do projecto, é possível utilizar esta biblioteca para consumir o WS.

Na Figura 86 é possível observar um excerto de código de uma aplicação Android que possibilita o consumo do referido WS. Como é possível verificar, é definido um conjunto de *Strings* (*namespace, url, soap_action* e *method_name*) que permitem identificar o serviço e a acção (método) a invocar. Isto deve-se ao facto de em Android não ser possível gerar automaticamente as classes necessárias para manipular os WS a partir do WSDL (é possível, por exemplo, em Java tradicional). Estas *Strings* são obtidas a partir da análise manual do WSDL. De seguida é criado um objecto do tipo "*SoapObject*", no qual são adicionadas *properties*, ou seja, os parâmetros de entrada do método a invocar. È criado o *envelope* SOAP e a conexão HTTP. A invocação do serviço é realizada através do método "*call()*". Este método é bloqueante, ou seja, aguarda que a resposta seja recebida. A resposta é obtida no objecto "*envelope*".

```
public class AndroidWebService extends Activity {

    private final String NAMESPACE = "http://www.webserviceX.NET/";
    private final String URL =
"http://www.webservicex.net/ConvertWeight.asmx";
    private final String SOAP_ACTION =
"http://www.webserviceX.NET/ConvertWeight";
    private final String METHOD_NAME = "ConvertWeight";

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        SoapObject request = new SoapObject(NAMESPACE, METHOD_NAME);
        String weight = "3700";
        String fromUnit = "Grams";
        String toUnit = "Kilograms";

        PropertyInfo weightProp =new PropertyInfo();
        weightProp.setName("Weight");
        weightProp.setValue(weight);
        weightProp.setType(double.class);
        request.addProperty(weightProp);

        PropertyInfo fromProp =new PropertyInfo();
        fromProp.setName("FromUnit");
        fromProp.setValue(fromUnit);
        fromProp.setType(String.class);
        request.addProperty(fromProp);

        PropertyInfo toProp =new PropertyInfo();
        toProp.setName("ToUnit");
        toProp.setValue(toUnit);
        toProp.setType(String.class);
        request.addProperty(toProp);

        SoapSerializationEnvelope envelope = new
SoapSerializationEnvelope(SoapEnvelope.VER11);
        envelope.dotNet = true;
        envelope.setOutputSoapObject(request);
        HttpTransportSE androidHttpTransport = new HttpTransportSE(URL);

        try {
            androidHttpTransport.call(SOAP_ACTION, envelope);
            SoapPrimitive response = (SoapPrimitive)envelope.getResponse();
            Log.i("myApp", response.toString());

            TextView tv = new TextView(this);
            tv.setText(weight+" "+fromUnit+" equal "+response.toString()+ "
"+toUnit);
            setContentView(tv);

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Figura 86 – Exemplo do consumo de um WS em Android